

# **SIGNAL DECOMPOSITION INTO PRIMITIVE KNOWN SIGNAL CLASSES**

by

David G. Galati

B.S. Electrical Engineering, University of Pittsburgh, 2000

Submitted to the Graduate Faculty of  
School of Engineering in partial fulfillment  
of the requirements for the degree of  
Master of Science

University of Pittsburgh

2002

UNIVERSITY OF PITTSBURGH

SCHOOL OF ENGINEERING

This thesis was presented

by

David G. Galati

It was defended on

April 3, 2002

and approved by

J. Robert Boston, Professor, Department of Electrical Engineering

Ching Chung Li, Professor, Department of Electrical Engineering

Thesis Advisor: Marwan Simaan, Professor, Department of Electrical Engineering

# Signal Decomposition Into Primitive Known Signal Classes

David G. Galati, M.S.

University of Pittsburgh, 2002

The detection of simple patterns such as impulses, steps, and ramps in signals is a very important problem in many signal processing applications. The human eye is a very effective filter and hence is capable of performing this task very efficiently. In applications where no humans are involved in the signal interpretation process, this task needs to be performed by a computer.

In this thesis, we propose and investigate two novel algorithms to automate this task. Our starting point is a discrete signal composed of an unknown number of ramps, steps, and impulses with unknown magnitudes and delays as well as random noise. We propose two different criteria based on “minimum energy” and “minimum complexity” to decompose the signal into these basic simple patterns. The solutions based on these criteria are proposed and examined. Over all, the “minimum complexity” criterion seems to produce results that are more similar to the human eye’s interpretation than the “minimum energy” approach.

## ACKNOWLEDGEMENTS

First I would like to thank Dr. Marwan Simaan of the University of Pittsburgh, without whose efforts and guidance, this endeavor would not have been possible.

I would also like to extend my thanks to Dr. J. Robert Boston and Dr. Ching-Chung Li for their efforts in serving on this committee.

This research was sponsored in part by the Defense Advanced Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL) under agreement number F33615-01-C-3151.

The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the DARPA, the AFRL, or the U.S. Government.

## TABLE OF CONTENTS

1.0	Introduction.....	1
1.1	Problem.....	3
1.2	Scope of Paper .....	9
2.0	Minimum Energy Multi Signal Deconvolution .....	11
2.1	Introduction.....	11
2.2	Derivation .....	11
2.2.1	Mathematical Characterization of Composite Signal .....	11
2.2.2	Representing Convolution as a Matrix Multiplication.....	14
2.2.3	Creating a Cost Function .....	15
2.2.4	Solving for the Control Signals.....	16
2.3	Testing.....	19
2.4	Conclusions About the Minimum Energy Approach.....	30
3.0	Minimum Complexity Multi Signal Deconvolution.....	32
3.1	Introduction.....	32
3.2	Derivation .....	32
3.2.1	Characterize Data Signal.....	32
3.2.2	Numeric Method Proposed .....	33
3.2.3	Assume Number, Types and Delays of Component Signals are Known .....	34
3.2.4	Finding the Signal Types and Delays .....	35
3.2.5	Determining the Minimum Number of Component Signals.....	36
3.2.6	Determining the Proper Threshold.....	37
3.2.7	Signal and Magnitude Constraints .....	43
3.3	Testing.....	43
3.4	Performance of Algorithm .....	48
4.0	Fast Minimum Complexity Algorithm .....	50
4.1	Introduction.....	50
4.2	Proposed Solution .....	50
4.3	Algorithm Parameters .....	51
4.3.1	Window Size.....	51
4.3.2	Window Increment.....	52
4.4	Performance Enhancement of Fast Algorithm.....	52
4.5	Algorithm Performance .....	54
4.6	Quantitative Noise analysis.....	56
4.6.1	The Impact of the Threshold on Performance .....	59

4.6.2	The Effect of Changing the Window Length.....	60
4.6.3	The Relation Between Error and Step Size.....	62
4.7	Conclusions.....	64
5.0	Final Comparison and Conclusion.....	71
5.1	End Goal of Paper.....	71
5.2	Thoughts on Minimum Energy Approach.....	71
5.3	Thoughts on Minimum Complexity Approach.....	72
5.4	Future Possibilities.....	73
Appendix A –	Algorithm Block Diagrams.....	74
Minimum Energy Approach.....		74
Original Minimum Complexity Algorithm.....		75
Fast Minimum Complexity Algorithm.....		76
Appendix B –	Sample Matlab Code.....	77
Minimum Energy Approach.....		78
Original Minimum Complexity Algorithm.....		82
Fast Minimum Complexity Algorithm.....		86
Bibliography.....		88

## LIST OF TABLES

Table 3.1 – M.C.A. Iterations for Selected Number of Component Signals and Data Lengths...	49
Table 4.1 – Comparison of Iterations for Several Data Lengths .....	54

## LIST OF FIGURES

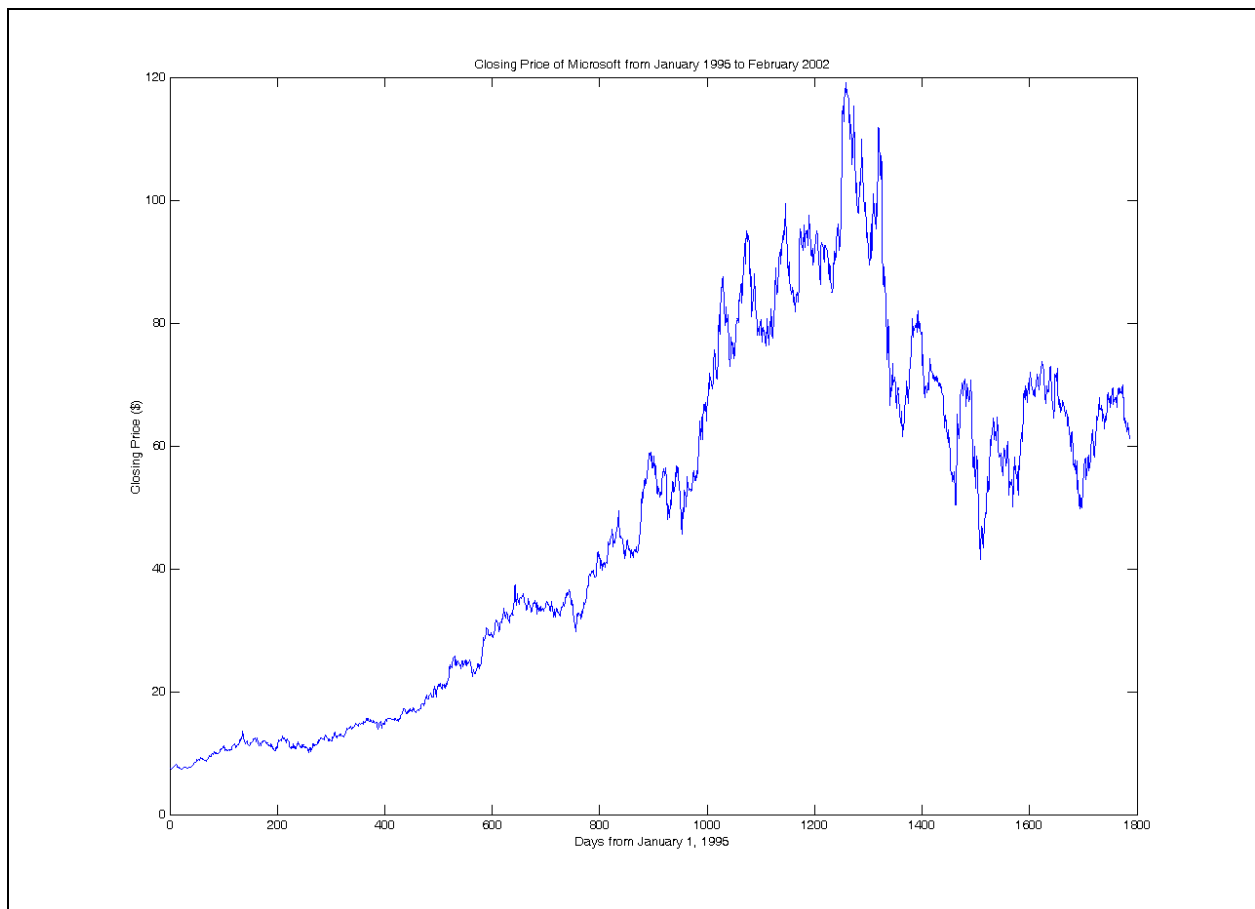
Figure 1.1 - Closing Price of Microsoft Over a Seven-Year Period.....	1
Figure 1.2 – Human Eye Interpretation of Microsoft Stock Data.....	3
Figure 1.3 – List of Component Signals .....	4
Figure 1.4 – Example of Troop Concentration .....	6
Figure 1.5 – Interpretation of Troop Concentration.....	6
Figure 1.6 – Two Examples of Composite Signals.....	8
Figure 2.1 – List of 3 Periodic Component Signals.....	20
Figure 2.2 – Results of Minimum Energy Approach When Considering Periodic Components.	21
Figure 2.3 – List of Uncorrelated Random Component Signals.....	23
Figure 2.4 – Results of Minimum Energy Approach When Considering Random Signals .....	24
Figure 2.5 – Illustration of Component Signals Being ‘Lost’ .....	25
Figure 2.6 – Effect of Noise on the Minimum Energy Approach .....	27
Figure 2.7 – Attempt at Noise Removal in the Minimum Energy Approach .....	28
Figure 2.8 – Set of Non-periodic, Cross-Correlated Component Signals.....	29
Figure 2.9 – Results of the M.E.A to Non-periodic Cross-Correlated Component Signals .....	30
Figure 3.1 – PDF’s of the Total Energy in White Noise Signals of Various Lengths.....	41
Figure 3.2 - CDF’s of the Total Energy in White Noise Signals of Various Lengths.....	42
Figure 3.3 – Results of the Minimum Complexity Algorithm to a Noise Free Signal .....	44
Figure 3.4 – Results of the M.C.A. a Signal with Small Additive Noise .....	45
Figure 3.5 – Results of the M.C.A. to a Signal with Moderate Additive Noise .....	46
Figure 3.6 - Results of the M.C.A. to a Signal with Large Additive Noise.....	47
Figure 3.7 – M.C.A. Iterations vs Signal Length.....	49
Figure 4.1 – Results of the F.M.C.A. on a Data Signal with No Additive Noise .....	55
Figure 4.2 – Results of the F.M.C.A. on a System With Additive Noise.....	56
Figure 4.3 – Relationship Between Noise Power and F.M.C.A. Performance.....	58
Figure 4.4 – Signal to Noise Ratio for F.M.C.A vs Threshold .....	59
Figure 4.5 – Signal to Noise Ratio for F.M.C.A. vs Window Size.....	61
Figure 4.6 – Signal to Noise Ratio for F.M.C.A vs Window Increment .....	63
Figure 4.7 – F.M.C.A. Applied to Noisy Signal .....	65
Figure 4.8 – F.M.C.A Applied to Very Noisy Signal .....	66



Figure 4.9 – FMCA Approximation of Microsoft Share Price (Long Term Trends) .....	67
Figure 4.10 - FMCA Approximation of Microsoft Share Price (Medium Length Trends).....	68
Figure 4.11 - FMCA Approximation of Microsoft Share Price (Short Length Trends).....	69
Figure A.1 – Minimum Energy Algorithm Flow Chart.....	74
Figure A.2 – Minimum Complexity Algorithm Flow Chart.....	75
Figure A.3 – Fast Minimum Complexity Algorithm Flow Chart.....	76

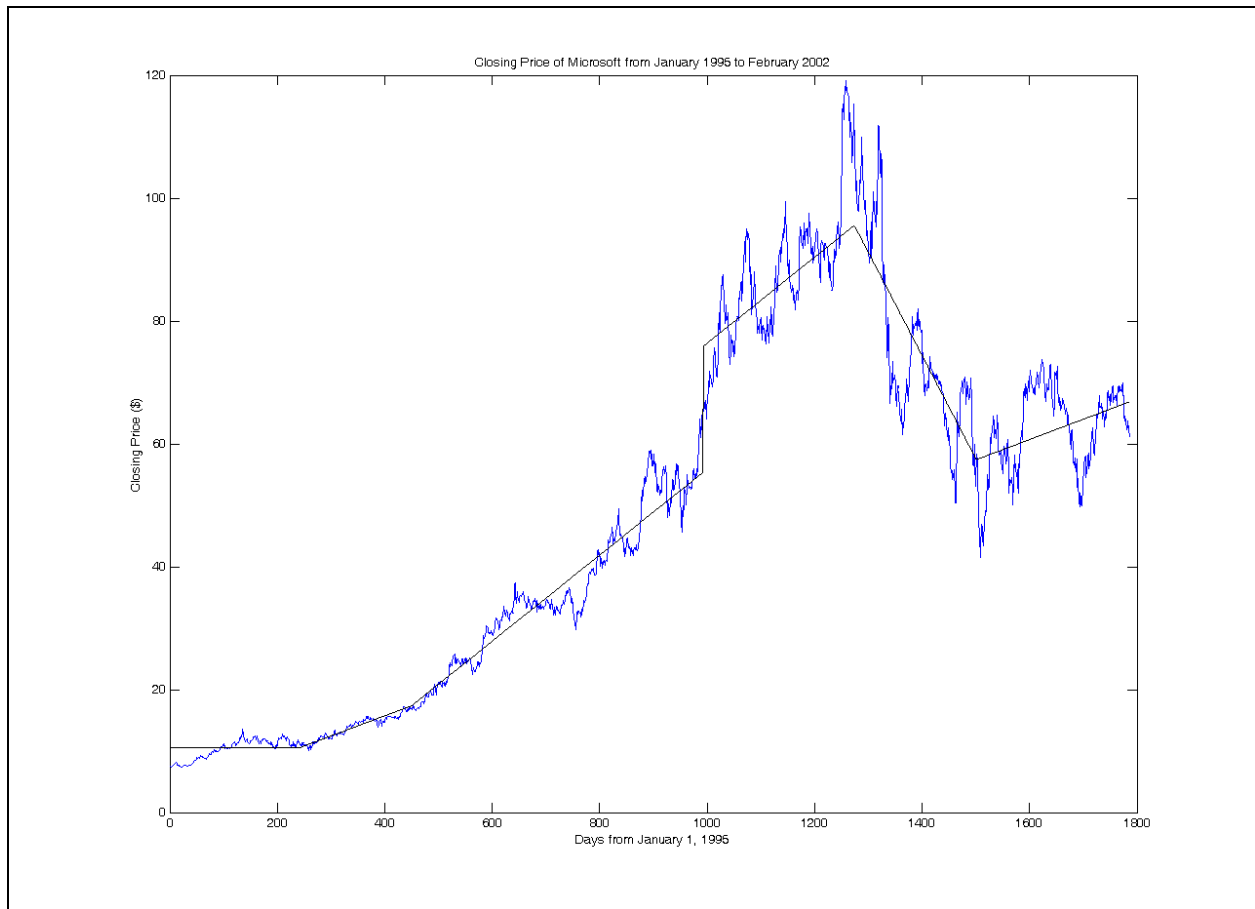
## 1.0 Introduction

When the human eye scans a set of data or an image, it shows a remarkable ability to segregate the information into regions. Consider for example, the closing price for a share of Microsoft stock from January 1995 to February 2002, as shown in Figure 1.1.



**Figure 1.1 - Closing Price of Microsoft Over a Seven-Year Period**

The human eye will instinctively draw a trend line over the data and divide it into sections. For example one interpretation might be that the share price gradually increased for the first 800 days at a roughly constant rate of about 2.8 cents a day. Over the next 200 days the stock price began to increase faster, at a rate of nearly 10 cents per day. At around day 1000 the stock price suddenly stepped up by around 20 dollars followed by 300 days of the same 10-cent a day increase. At day 1300 a significantly negative event occurred and Microsoft's stock price dropped by more than 40\$ a share. Trust was never regained in the stock and for the remaining 400 days the stock price remained essentially constant. Another such interpretation is illustrated in Figure 1.2. Such a quick human eye interpretation will be referred to as an intuitive solution. Clearly the human eye has ignored the random fluctuations in price and concentrated on the simple or primitive patterns of price increase, decrease, and sudden price jumps.

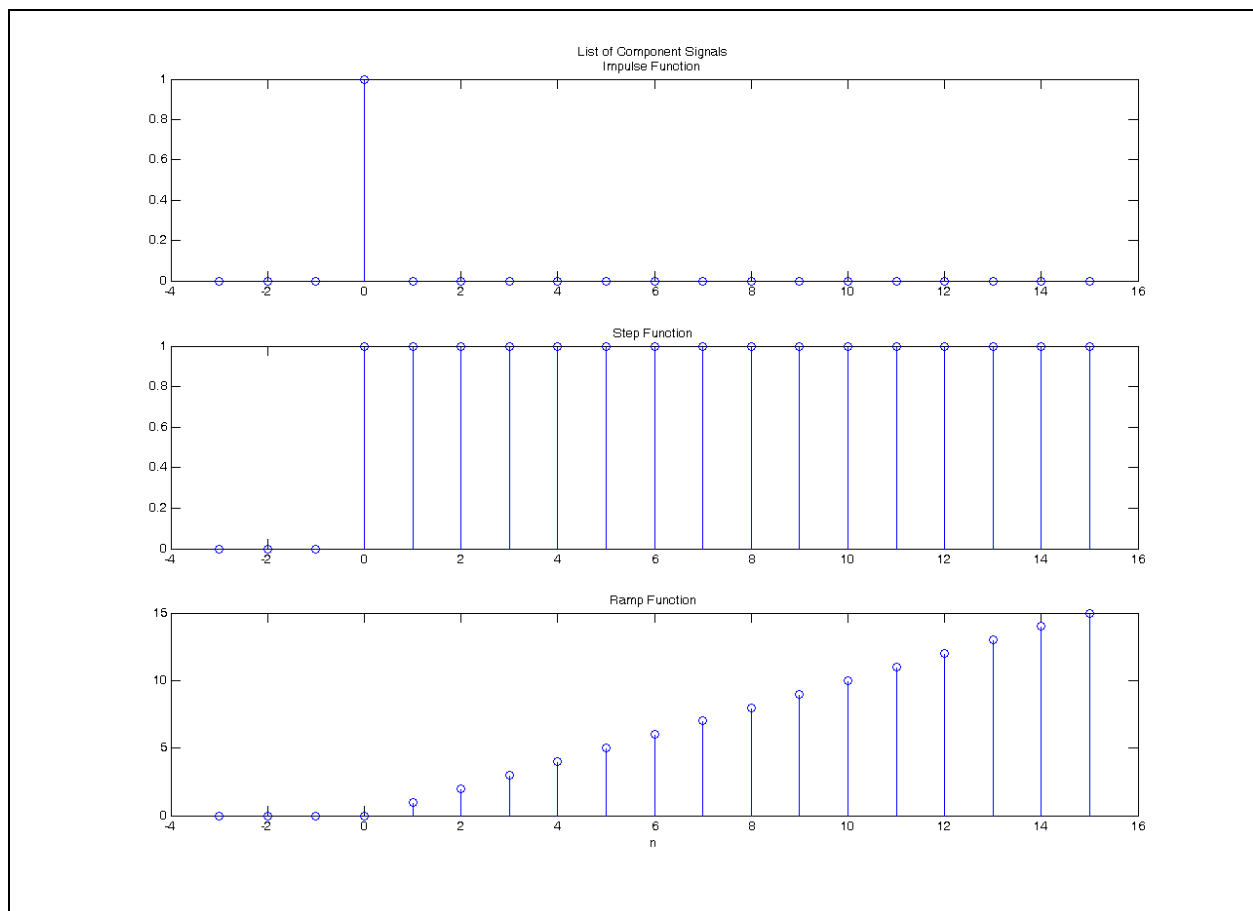


**Figure 1.2 – Human Eye Interpretation of Microsoft Stock Data**

## **1.1 Problem**

Many situations exist in which a signal is formed as a combination of other known simple signals. On occasion, it is desirable to decompose this signal back to its component signals. While this may appear to be a simple problem when the component signals are known, it is a completely different case when all that is known is the possible types of component signals. Making the problem still more difficult is the possibility of exposure to additive noise.

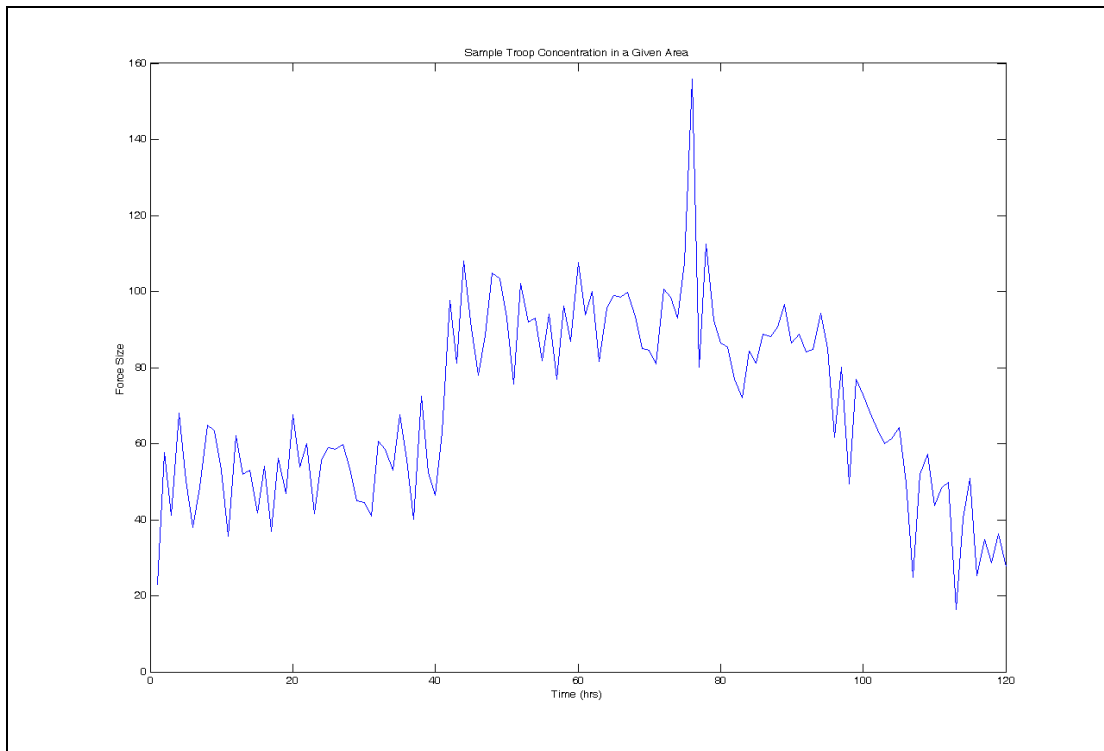
Typical examples of such signals that we are interested in, in this thesis, are those composed of primitive signals, such as impulses, steps, and ramp functions. Many real world signals can be modeled as resulting from the sum of various shifted and scaled versions of primitives. The set of primitives consisting of ramps, steps, and, impulses can be used very frequently to model real world systems. These primitives, shown in Figure 1.3, appear often and are easily separated by the human eye.



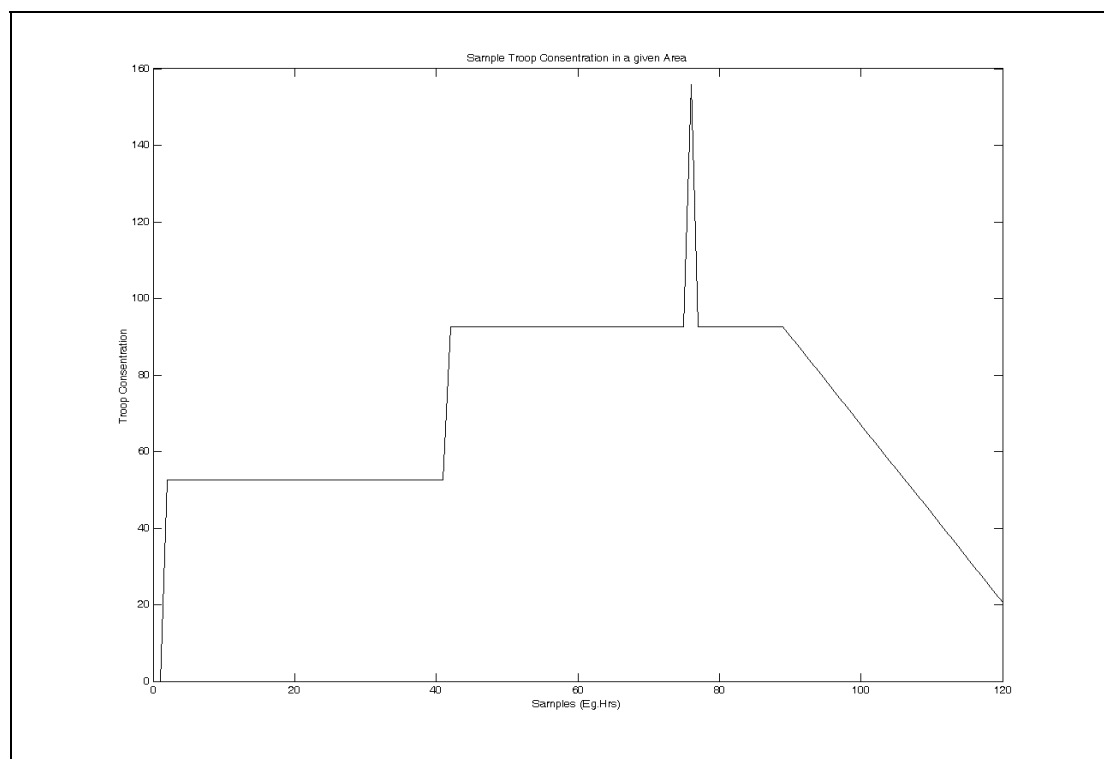
**Figure 1.3 – List of Component Signals**

In this thesis, we are interesting in automating this process. We consider a signal that consists of ramps, steps, and impulses occurring at unknown times in unknown numbers with unknown magnitudes. A method is desired to automate the process of decomposition while also insuring that the decomposition will agree with a human interpretation. This does not mean that the solution has to exactly agree with any human interpretation of the data, as different people will decompose the signal differently. The process must only provide a solution that could be generated by a human.

There are many other signal processing applications where such a method would prove useful. For example, traffic on a certain stretch of road could consist of step increases in traffic at rush hour and a quick spike in volume at lunch. Other increases and decreases in traffic might occur more slowly throughout the course of the day. If one wanted to know what the traffic pattern looked like on average, and only a limited amount of data was available, signal decomposition would yield the only truly accurate results. Filtering the signal with linear filters would yield smoothed or blurred results. The traffic from a plant letting out at 5pm would appear as a gradual increase, instead of an all out rush to exit the parking lot. Another example would be troop concentration in a battle [16], such as the example given in Figure 1.4. Sensors do not give completely accurate results. A sensor will add a certain amount of noise to the actual troop concentration. If the intelligence is poor, as in the case of Figure 1.4, transitions in the levels of troop concentration are difficult to detect. If the enemy dramatically increased the number of soldiers in a given area, the ability to quickly detect this event would be very helpful to the planning of a future military maneuver.



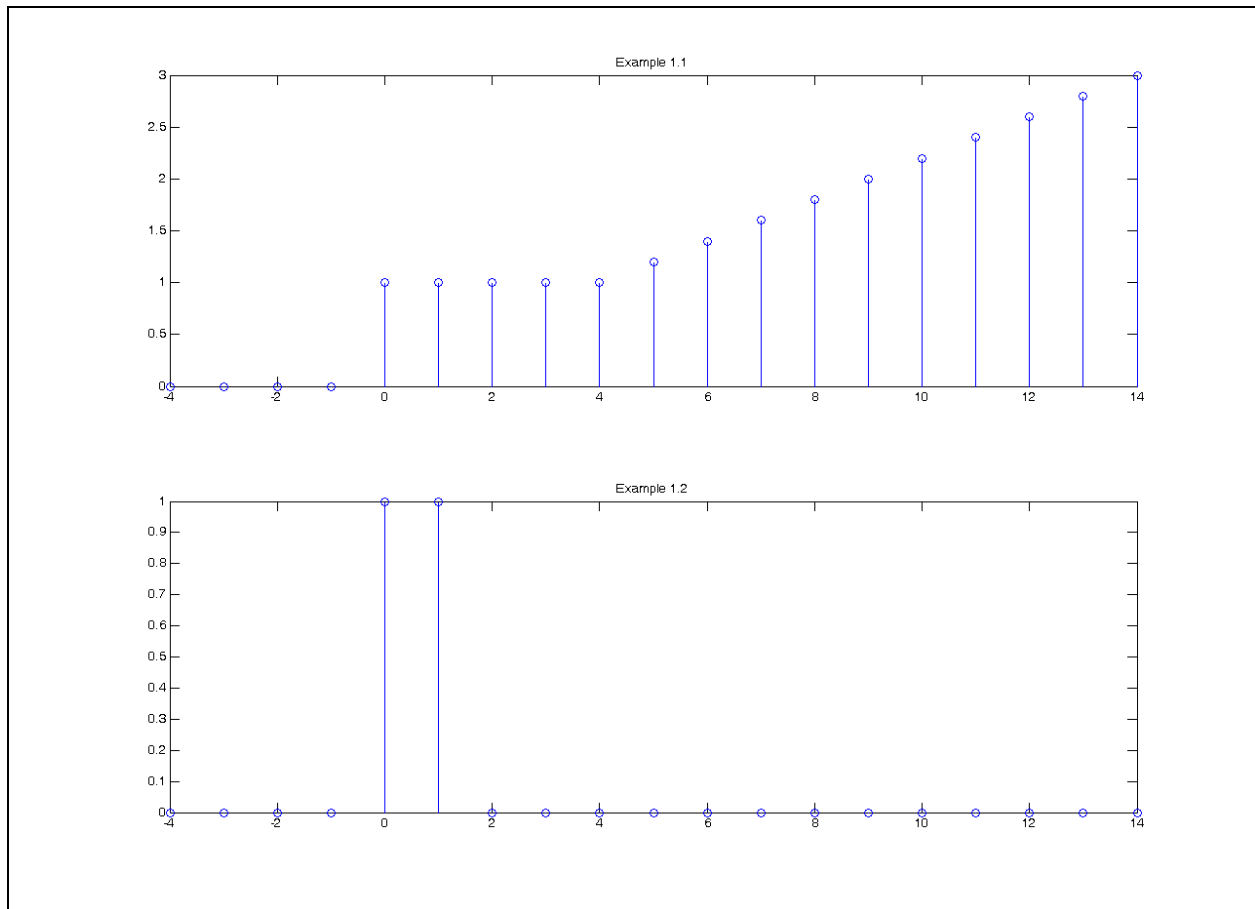
**Figure 1.4 – Example of Troop Concentration**



**Figure 1.5 – Interpretation of Troop Concentration**

It is very important that the algorithm is able to detect the step at  $t=40$  up followed by a sharp spike and a ramp down in concentration at times 75 and 85 as detected in Figure 1.5. This information is vital in planning future battles. As a result, a method is desired to separate a composite signal into a combination of shifted and scaled component signals. In most systems of this type, there are an infinitive number of possible combinations of component signals that could be combined to create the composite signal. Because not all of the possible combinations of component signals that yield the composite are desirable, the ‘correct’ solution set of components signals must satisfy at least one other requirement. For the method sought after in this thesis, only one additional requirement is considered. The method of separating a signal into its components must yield a result that could be generated by a human eye. We refer to such solutions as intuitive interpretations. This means that when looking at a simple composite signal, particularly a signal free of noise, the instantly recognizable solution is the intuitive result. For example, looking at the composite signal in Figure 1.6 (a), the intuitive solution is that the signal is composed of a step function of magnitude 1 at the origin and a ramp of slope .2 beginning at point 4.





**Figure 1.6 – Two Examples of Composite Signals**

Again however, even though the intuitive solution would be considered the correct answer it is not the only possible combination of component signals that could result in Figure 1.6 (a). An infinite number of combinations of the components could produce such a result. For example, the composite signal could be decomposed into a series of impulses, or a series of step functions, or even a combination of the two. This makes the intuitive solution difficult to find among all the possible solutions without some mathematical measure of the intuitive solution.

This point is better illustrated in Figure 1.5 (b). Here, instead of one definitive solution answer there are two solutions that satisfy the intuitive criterion. It could be argued that the

signal is composed of two impulses. It could also be decomposed into a step followed by a second of equal but opposite magnitude two points later, with equal validity. Although these would be considered valid intuitive answers decomposing the signal into two steps and two impulses would be considered unintuitive. This begs the question, what exactly makes the intuitive answer intuitive? Of course a part of the solution must lie in minimizing the error between the sum of the composite signals and the actual signal, but many solutions that satisfy this requirement exist.

This problem is unlike many problems that have been solved in the past. Various Prony methods [1-6], the Principal Component Method [7,6], and the Pisarenko [8-10,6] and Music [11,6] methods bear some similarities to this problem. These deal with a problem known as modal analysis. In this method a signal is assumed to be composed of a fixed number of damped cosines of unknown damping, magnitude, and phase.

Unfortunately, these solutions require the component signals to be of the form of a damped cosine. What is needed is a solution to the more general problem, allowing the signals to be of any arbitrary form. Furthermore, because an infinite number of solutions to this problem occur, the solution must agree with the intuitive result.

## **1.2 Scope of Paper**

In order to systematically find a solution that consistently agrees with the intuitive answer we must first mathematically characterize this unknown requirement. Two approaches will be looked at and compared in the following chapters. The first, a minimum energy approach, treats the composite signal as a multi-signal deconvolution problem. Using optimization approaches a closed form solution will be derived and examined. In the second approach a concept of minimum complexity will be examined. In this method the number of signal components is

minimized. These two approaches will then be compared and characterized to gain some insight into the nature of the intuitive solution.

## **2.0 Minimum Energy Multi Signal Deconvolution**

### **2.1 Introduction**

The first criteria considered for the decomposition of a signal into its individual components is that of a minimum energy principle. This is a very useful principle not only because it essentially eliminates all but one solution but also it is a well-defined concept. This is the approach used when decomposing a composite signal into a set of component signals if the number, delay, and type of component signals are known. A great deal of work has been done devising methods to solve minimum energy type problems [12]. As a result, it is only natural to attempt to extend this theory to a more general case.

### **2.2 Derivation**

#### **2.2.1 Mathematical Characterization of Composite Signal**

In order to find the minimum energy solution the problem must be written in a form that allows a minimum energy approach. A signal model must be defined before it can have its energy minimized. First, it will be assumed that there exists a system that creates an  $N$  point composite signal  $y$ . This signal is created from a combination of a set  $n$  types of component signals, each  $N$  points long. These component signals are controlled by a second set of signals also of length  $N$ . The control signals will control the magnitude and the delay of the component signals being added into the composite signal. For example if a component signal type is defined as a step function of magnitude 1 and a composite signal is made up of the sum of step functions of magnitude 3 at points 1, 5, and 8, the control signal would be composed of impulses of magnitude 3 at points 1, 5, and 8 and zeros everywhere else.

Now because we desire a minimum energy type solution, the natural choice is to minimize the total energy contained in the control signals. To this end each of the component signals  $c_i$  is assigned a controlling signal  $s_i$ . The easiest way to represent the relationship between the control signals and the component signals is to use convolution.

$$z_n = \sum_{k=0}^{N-1} x_k y_{n-k} \quad (2.2.1)$$

This is made apparent if  $x$  is instead represented by a series of delta functions

$$x_n = 2\delta_{n-1} + 5\delta_{n-5} \quad (2.2.2)$$

Substituting  $x$  back into the definition of convolution provides the following familiar result.

$$\begin{aligned} z_n &= \sum_{k=0}^{N-1} (2\delta_{k-1} + 5\delta_{k-5}) y_{n-k} \\ z_n &= 2y_{n-1} + 5y_{n-5} \end{aligned} \quad (2.2.3)$$

If  $x$  is assumed to consist of a series of delta functions then the resulting convolution will provide a sum of shifted and scaled copies of the signal  $y$ . Because this is exactly the relationship between the control signal and its corresponding component signal on the composite output, convolution proves to be a logical mathematical representation, with however, one problem.

In convolution, when two signals of length  $N$  are convolved together the resulting signal is of length  $2N-1$ . This causes a problem because the component signal is limited to a length of only  $N$ . An obvious solution would be to limit the component signal and its control signal to a total combined length of  $N+1$ . This also causes many problems. For a component signal to exist along the entire length of the composite signal it must be of length  $N$ . In addition, for the component signal to be shifted to the last point in the composite signal the control signal must also be of length  $N$ . The only solution to this problem is to make both the component signal and the corresponding control signal a length of  $N$ . When these two vectors are convolved, they will

result in a signal of length  $2N-1$ . Fortunately, only the first  $N$  points are important. The remaining points can be truncated.

It is now possible to represent the composite signal mathematically. It is assumed to be composed of a sum of convolutions. Each component signal will be convolved with its corresponding control vector. The resulting convolutions will be added and then truncated. Expressed mathematically, the composite vector  $y$  can be assumed to be as follows,

$$y_k = \sum_{i=1}^n \left( s_{(i)k} * c_{(i)k} \right) \Big|_{k=0}^{N-1} \quad (2.2.4)$$

where the notation  $\hat{y}_k = y_k \Big|_{k=0}^{N-1}$  is used to indicate that  $\hat{y}_k$  consists of only the 1<sup>st</sup>  $N$  samples of  $y_k$ .

Because this composite signal is eventually measured, additive noise must be considered. Therefore as part of the system model it is assumed that a noise signal of power  $\sigma$  is added to the actual component signal  $y_k$ .

$$\tilde{y}_k = y_k + \sigma \cdot \eta_k \quad (2.2.5)$$

Now because  $y(k)$  contains additive noise, an estimate for the composite signal must be defined. This is because when noise is introduced, it is not always possible to exactly reconstruct the original signal. Similar to the definition of  $y(k)$  it differs only in that it assumes that all of the  $s(k)$  signals can only be estimated.

$$\hat{y}_k = \sum_{i=1}^n \left( \hat{s}_{(i)k} * c_{(i)k} \right) \Big|_{k=0}^{N-1} \quad (2.2.6)$$

## 2.2.2 Representing Convolution as a Matrix Multiplication

Working with a truncated convolution can make the calculations very difficult. As a result a simpler expression is required. To simplify this type of operation we will consider a convolution of two signals, each of length four as follows.

$$\begin{aligned}x_n &= \{x_0, x_1, x_2, x_3\} \\y_n &= \{y_0, y_1, y_2, y_3\} \\z_n &= x_n * y_n \\z_n &= \sum_{k=0}^3 x_k y_{n-k}\end{aligned}\tag{2.2.7}$$

Writing out the convolution sum leaves the following expression.

$$z = x_0 y_n + x_1 y_{n-1} + x_2 y_{n-2} + x_3 y_{n-3}\tag{2.2.8}$$

Because the functions are defined only in the space from zero to three they are assumed to be zero outside this space. With this in mind it is possible to solve for each value of  $z(n)$ .

$$\begin{aligned}z_0 &= x_0 y_0 \\z_1 &= x_0 y_1 + x_1 y_0 \\z_2 &= x_0 y_2 + x_1 y_1 + x_2 y_0 \\z_3 &= x_0 y_3 + x_1 y_2 + x_2 y_1 + x_3 y_0 \\z_4 &= x_1 y_3 + x_2 y_2 + x_3 y_1 \\z_5 &= x_2 y_3 + x_3 y_2 \\z_6 &= x_3 y_3\end{aligned}\tag{2.2.9}$$

Note that the convolution of two vectors of lengths  $L_1$  and  $L_2$  will result in a vector of length  $L_1+L_2-1$ , in this case 7. However, we desire only the first  $n$  values of the convolution, leaving the following resultant.

$$\begin{aligned}z_0 &= x_0 y_0 \\z_1 &= x_0 y_1 + x_1 y_0 \\z_2 &= x_0 y_2 + x_1 y_1 + x_2 y_0 \\z_3 &= x_0 y_3 + x_1 y_2 + x_2 y_1 + x_3 y_0\end{aligned}\tag{2.2.10}$$

This can be written in matrix form as follows.

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} x_0 & 0 & 0 & 0 \\ x_1 & x_0 & 0 & 0 \\ x_2 & x_1 & x_0 & 0 \\ x_3 & x_2 & x_1 & x_0 \end{bmatrix} \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} \quad (2.2.11)$$

This result can be generalized to represent any n truncated convolution of two vectors of size N as follows [13].

$$\left( \hat{s}_{(i)k} * c_{(i)k} \right) \Big|_{k=0}^{N-1} = Q_i c_i \quad (2.2.12)$$

Where Q is defined as the following diagonally symmetric matrix.

$$Q_i = \begin{bmatrix} c_0 & 0 & 0 & \dots & 0 \\ c_1 & c_0 & 0 & \ddots & \vdots \\ c_2 & c_1 & c_0 & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ c_{N-1} & \dots & c_2 & c_1 & c_0 \end{bmatrix} \quad (2.2.13)$$

Now applying this transform to the initial equation results in the following formulation of the data signal. Here each Q is an NxN matrix uniquely created by a single component signal in the given set of component signals and each possessing a corresponding Nx1 control signal, c.

$$y(k) = \sum_{i=1}^n Q_i s_i \quad (2.2.14)$$

### 2.2.3 Creating a Cost Function

Because we desire a minimum energy approach and most minimum energy problems are solved using optimization techniques, the most logical starting point would be to phrase the problem in terms of a cost function. This cost function must minimize the total energy contained in the set of control vectors s(k) and also minimize the difference between the estimated



composite signal and the noisy composite signal. The following equation is a cost function satisfying these requirements written using inner product notation.

$$C(s, x, \tilde{y}) = \sum_{i=1}^n \langle s_i, s_i \rangle + \langle \tilde{y}, \tilde{y} \rangle \quad (2.2.15)$$

Substituting in for the error term yields

$$C(s, x, \tilde{y}) = \sum_{i=1}^n \langle s_i, s_i \rangle + \left\langle \sum_{i=1}^n Q_i s_i - y, \sum_{i=1}^n Q_i s_i - y \right\rangle \quad (2.2.16)$$

Although this expression appears to be an appropriate cost function further analysis hints otherwise. Because the cost function is unconstrained, minimizing the energy of the controlling signals is equivalent to minimizing the error between the approximated signal and the noisy signal. Now although this could be remedied by a scale factor it shows a disturbing phenomenon. It is impossible to absolutely minimize both the error and the energy at the same time. The absolute minimum energy occurs when the control signals are set to zero while the absolute minimum error requires some energy in the control signals. As a result the energy and the error will be held in tension, the outcome dependent on an arbitrary scale factor. To solve this problem we will first assume that  $\sigma$ , the noise variance, is zero, forcing  $y(k)$  and its estimate to be equal.

$$\hat{y}(k) = y(k) \quad (2.2.17)$$

#### 2.2.4 Solving for the Control Signals

This greatly simplifies our initial set up of the cost function as it can be assumed that there will be no error between the actual function and the estimated function. The problem can then be constructed as a constrained optimization problem, removing the trouble caused by balancing the error with the minimum energy. Constructing a cost function that minimizes the energy of the control signals while using the above equation as a constraint results in the following set of necessary conditions.

$$J(s, \lambda) = \frac{1}{2} \sum_{i=1}^n \langle s_i, s_i \rangle + \lambda' \left( \sum_{i=1}^n Q_i s_i - \tilde{y} \right) \quad (2.2.18)$$

$$\nabla_{s_i} J(s, \lambda) = s_i + Q_i' \lambda = 0 \quad (2.2.19)$$

$$\nabla_{\lambda} J(s, \lambda) = \sum_{i=1}^n Q_i s_i - \tilde{y} = 0 \quad (2.2.20)$$

By rewriting (2.2.19) we obtain the following expression.

$$s_i = -Q_i' \lambda \quad (2.2.21)$$

Substituting (2.2.21) into (2.2.20) yields

$$\sum_{i=1}^n Q_i Q_i' \lambda = -\tilde{y} \quad (2.2.22)$$

Now because any matrix of rank n multiplied by it's transpose is guaranteed to be positive definite and also because two positive definite matrixes will always add up to a positive definite matrix the summation in (2.2.22) will always have an inverse.

$$\lambda = \left( \sum_{i=1}^n Q_i Q_i' \right)^{-1} - \tilde{y} \quad (2.2.23)$$

Because we now have an expression for lambda substituting (2.2.23) back into (2.2.21) gives a unique solution for each control vector.

$$s_j = (Q_j') \left( \sum_{i=1}^n Q_i Q_i' \right)^{-1} \tilde{y} \quad (2.2.24)$$

In order to gain some understanding of exactly what this solution entails we need to conduct a closer examination of the summation that is inverted. Looking at each QQ' pair in this summation and assuming that they were created from a vector  $\{c_0, c_1, \dots, c_j\}$ , the following matrix is obtained.

$$Q_i Q_i' = \begin{bmatrix} c_0^2 & c_0 c_1 & c_0 c_1 + c_0 c_2 & \cdots & \sum_{i=1}^j c_0 c_i \\ c_0 c_1 & c_0^2 + c_1^2 & & & \sum_{i=1}^j c_1 c_i \\ c_0 c_1 + c_0 c_2 & & c_0^2 + c_1^2 + c_2^2 & & \sum_{i=1}^j c_1 c_i \\ \vdots & & & \ddots & \vdots \\ \sum_{i=1}^j c_0 c_i & \sum_{i=1}^j c_1 c_i & \sum_{i=1}^j c_1 c_i & \cdots & \sum_{i=1}^j c_i^2 \end{bmatrix} \quad (2.2.25)$$

In this matrix, the  $k^{\text{th}}$  diagonal term corresponds to the energy in the signal  $c$  up to sample  $k$ . The off diagonal terms are representations of the autocorrelation of the signal up to sample  $k$ .

We now have a simple closed form solution to the minimum energy approach without noise; however, a solution to the signal with noise added is still required. To solve this problem a simple trick is proposed. Recall the equations that set up this problem in the beginning of the chapter.

$$\begin{aligned} y &= \sum_{i=1}^n (s_i * c_i) \Big|_{k=0}^{N-1} \\ \tilde{y} &= y + \sigma \cdot \eta \\ \hat{y} &= \sum_{i=1}^n (\hat{s}_i * c_i) \Big|_{k=0}^{N-1} \end{aligned} \quad (2.2.26)$$

If the noise term is assumed to be convolved with a delta function the following expression can be reached.

$$\tilde{y} = \sum_{i=1}^n (s_i * c_i) \Big|_{k=0}^{N-1} + \sigma \cdot \eta * \partial(k) \quad (2.2.27)$$

Note that because convolving with a delta function leaves the signal unchanged it is possible to move the noise factor into the summation. In order to do this an additional component vector has to be added, that of  $\sigma \delta(k)$ . Of course the noise power is not always exactly known, however various methods exist to help determine this quantity. However, for simplicity, in this

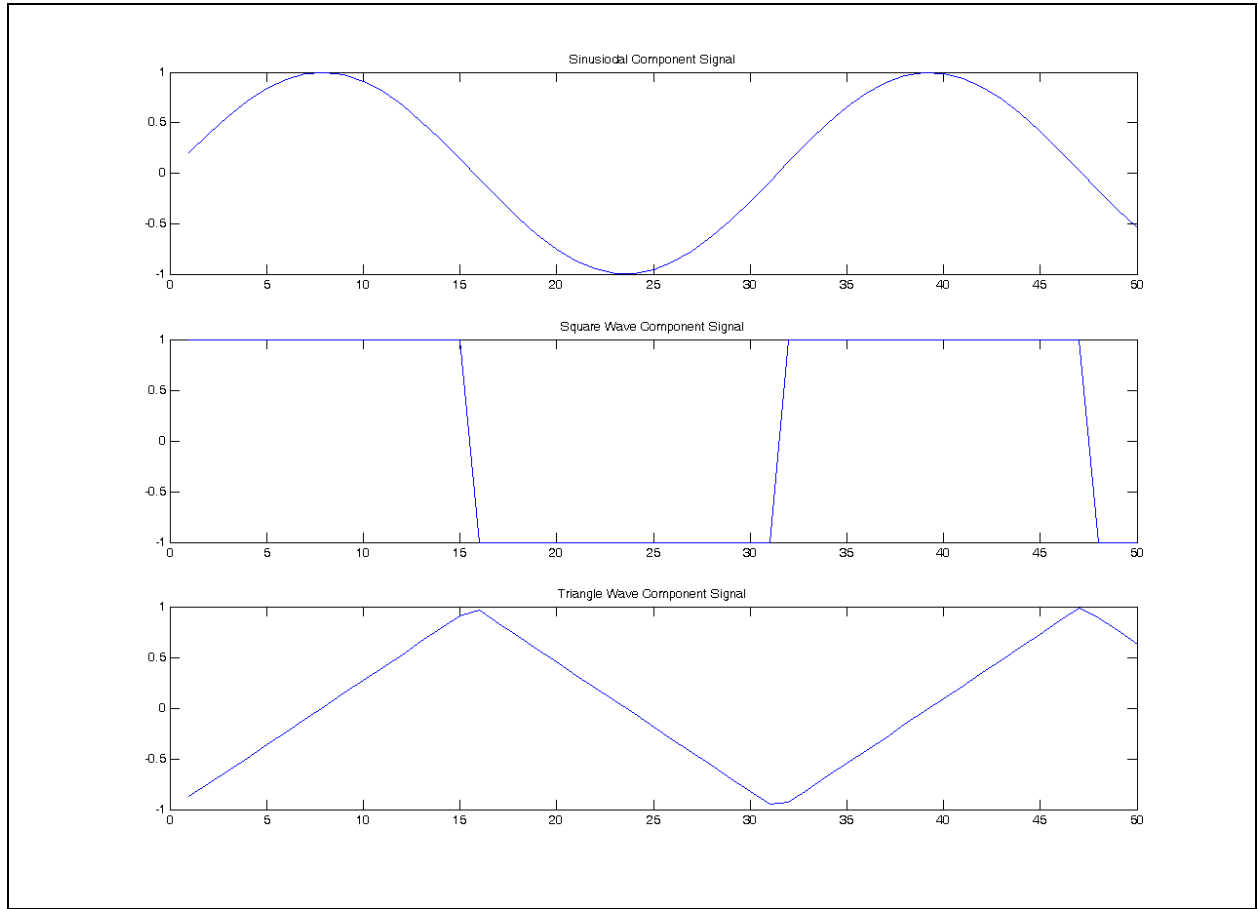
chapter we shall assume that this quantity is known exactly. By appending the extra component and assuming the error signal is the  $n+1^{\text{th}}$  control signal the noisy composite signal can be written as follows.

$$\tilde{y}(k) = \sum_{i=1}^{n+1} (s_i * c_i) \Big|_{k=0}^{N-1} \quad (2.2.28)$$

It can now be assumed that no additional error is present in the system and the assumption made in assuming zero noise can now be considered valid.

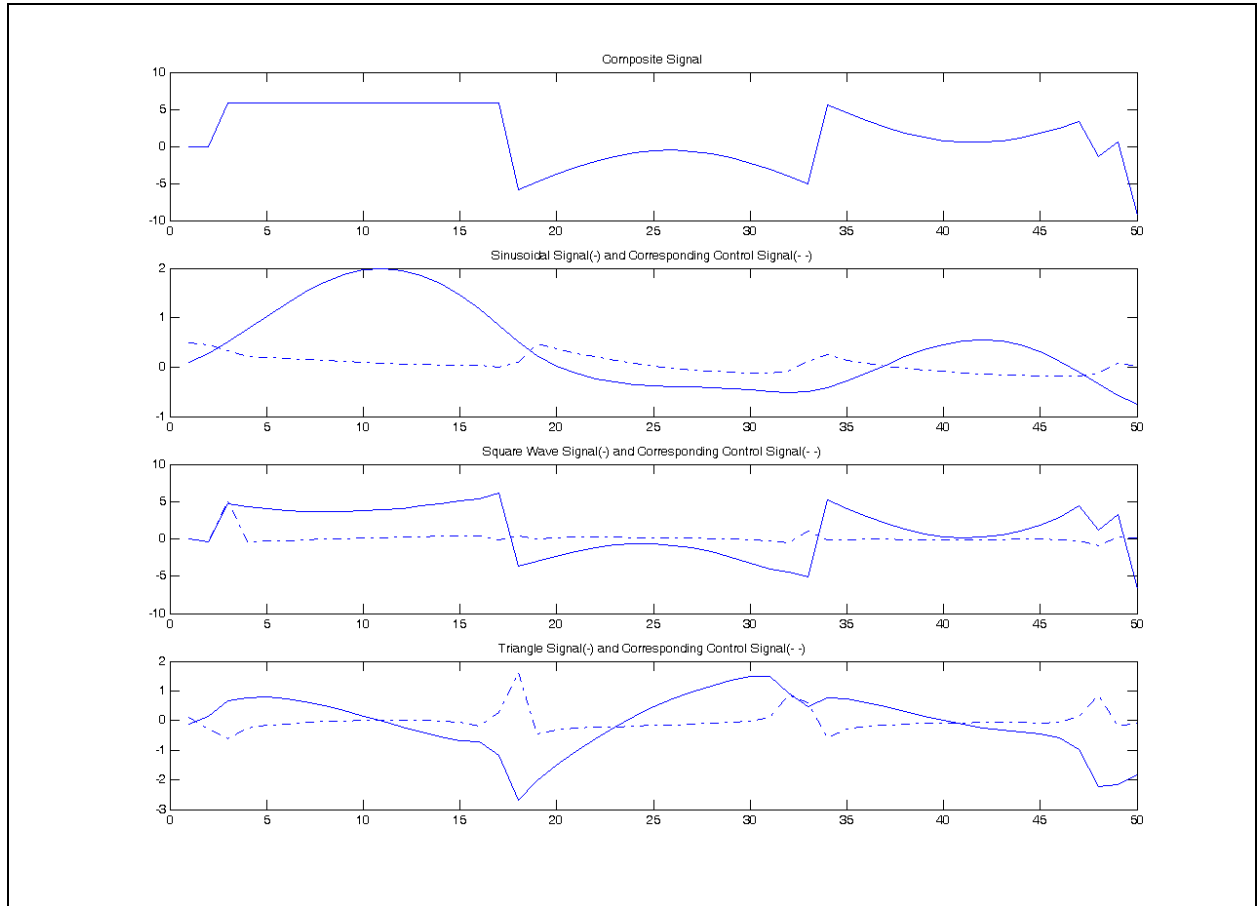
### 2.3 Testing

Armed with a final result we are now able to test the usefulness of the minimum energy approach in terms of signal decomposition. For the first test three similar signals will be used as the set of component signals. The array of component signals will be composed of a sinusoid, a square wave, and a triangle wave as shown in Figure 2.1.



**Figure 2.1 – List of 3 Periodic Component Signals**

Using this set of components a composite signal is created. This is done by shifting and multiplying each of the three component signals by a random amount. Then all of the shifted and scaled signals are added together producing a composite signal. The algorithm can then be applied to this composite signal and the set of components producing a set of control vectors. Figure 2.2 shows each control signal with a broken line and the result of the convolution of each component signal with its corresponding control signal as a continuous line.

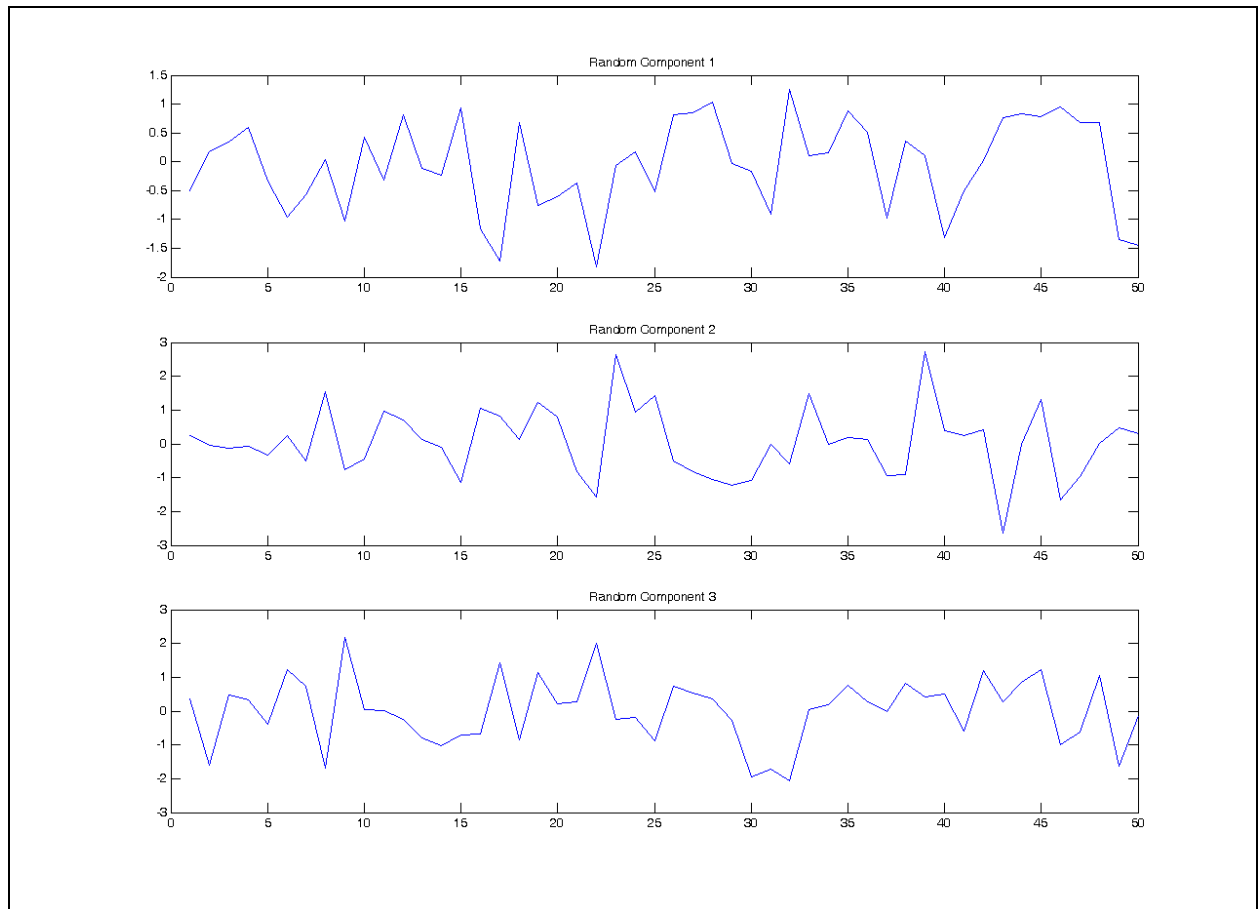


**Figure 2.2 – Results of Minimum Energy Approach When Considering Periodic Components**

Looking at the results even though the control signals combine to generate an exact match for the composite signal it is clear that the control signals bear little resemblance to the ones used to generate the signal. The sinusoidal component may look like a sinusoid but it appears to be completely out of phase with the expected result. Having a much better result, the square wave appears to roughly approximate the correct square wave, and if it was passed through a dead zone filter it would make a fairly accurate representation of the assumed answer.

Rounding out the three, the triangle wave appears the worst of all, looking completely unlike a triangle wave. Of great interest in this example is the difference in energy between the control signals that were used to create the signal and the ones that the algorithm generated. The original control signals have a combined energy of around 106 while the algorithms control signals have only a energy measure of 38, considerably less than the initial control signals.

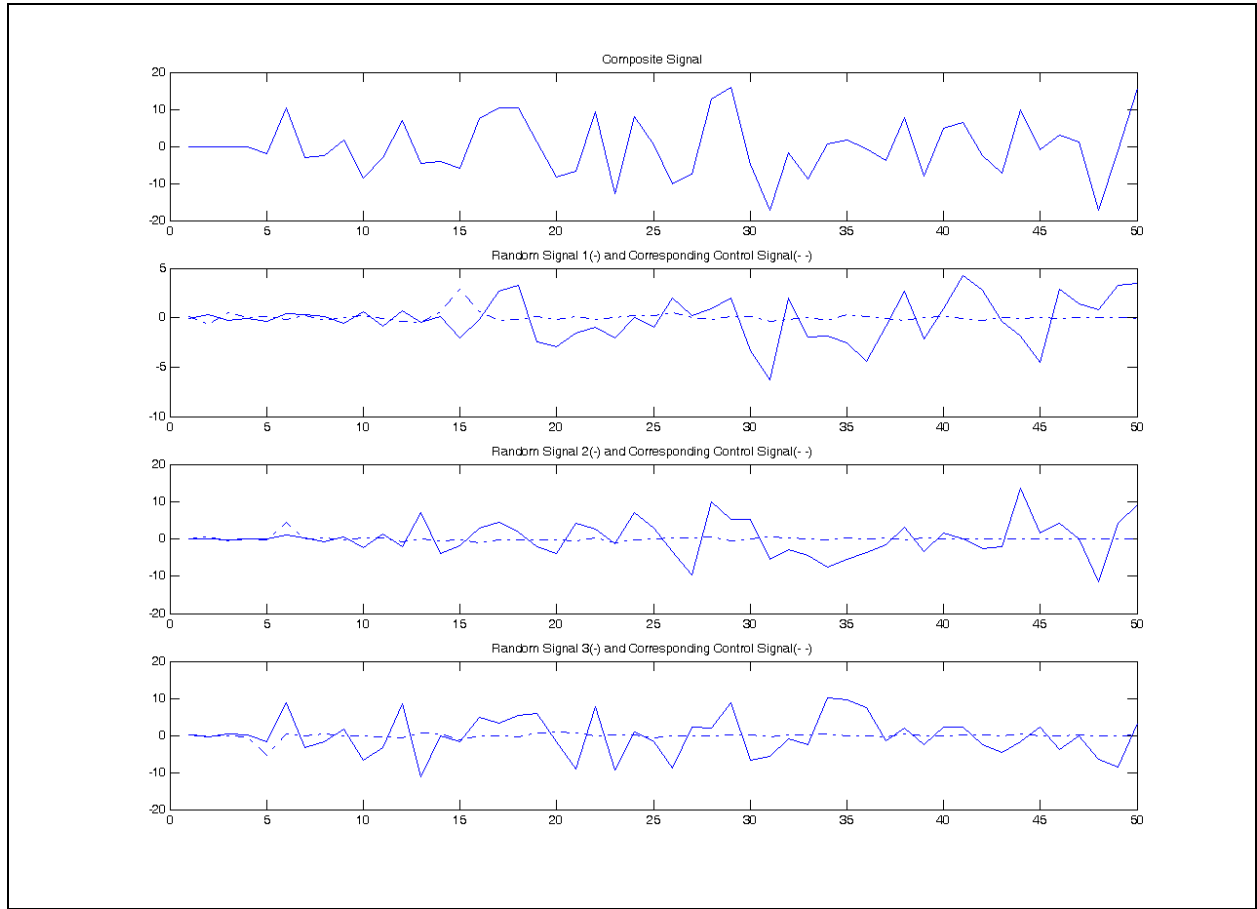
One possible explanation for the results obtained with the previous example is that the poor performance was due to strong correlations in the set of component signals. To determine the validity of this argument a trial was set up where each of the components signals was constructed from samples of independent Gaussian white noise. The component signals used are listed in Figure 2.3. Because these signals are Gaussian and white they are assured to be a virtually uncorrelated set of components.



**Figure 2.3 – List of Uncorrelated Random Component Signals**

In the same fashion as in the previous example a composite signal was created, after which the algorithm was applied. The results are listed in Figure 2.4.

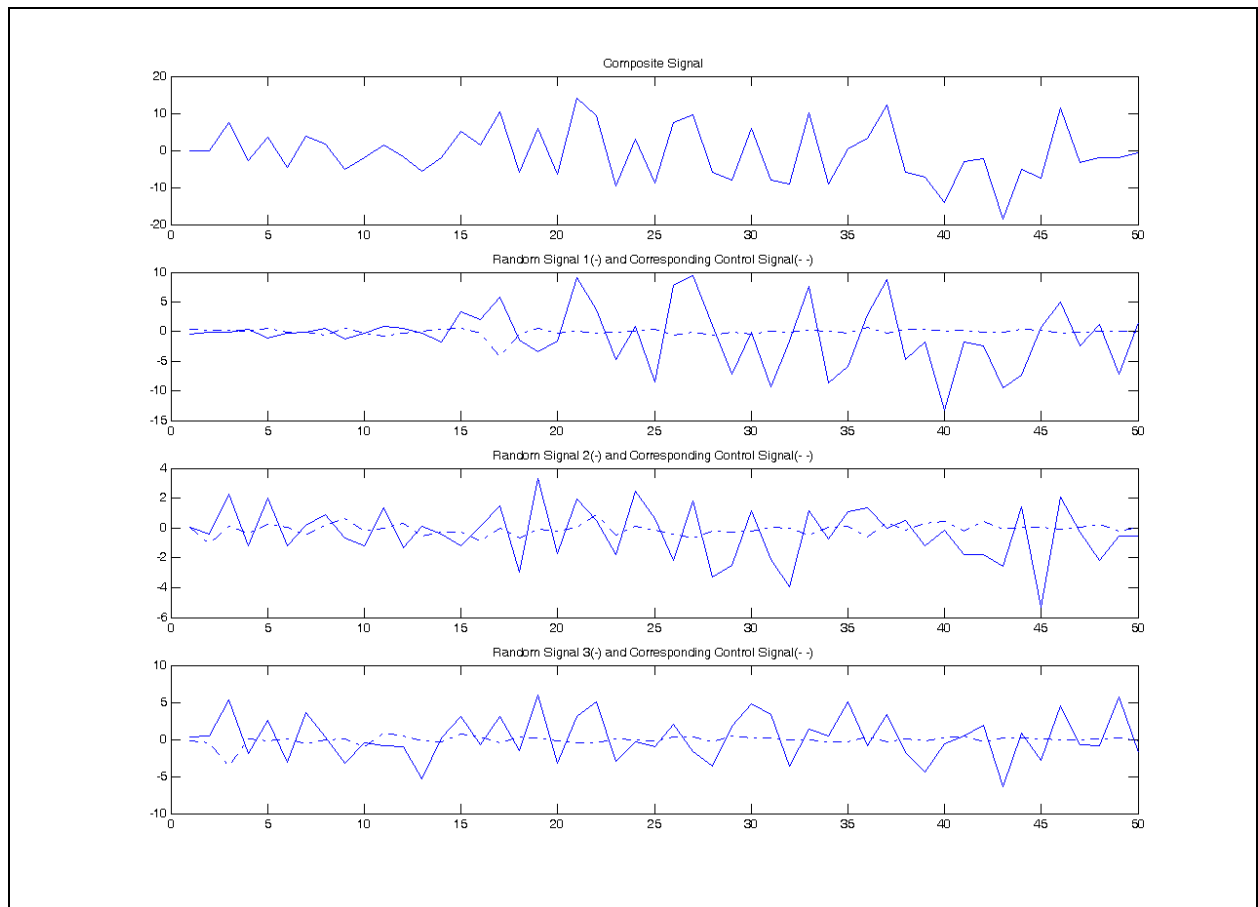




**Figure 2.4 – Results of Minimum Energy Approach When Considering Random Signals**

Although it is not possible to see an intuitive answer due to the fact that all of the signals are simply noise, when looking at the control signals, the characteristics of the intuitive result are apparent. Each of the three control signals has essentially one dominant peak located at the location of the peak in the original control signal. In addition, the energy content is much more similar to the original control signals. The original control signals have an energy content of 104 and the calculated control signals have an energy of around 73. After running this simulation a few

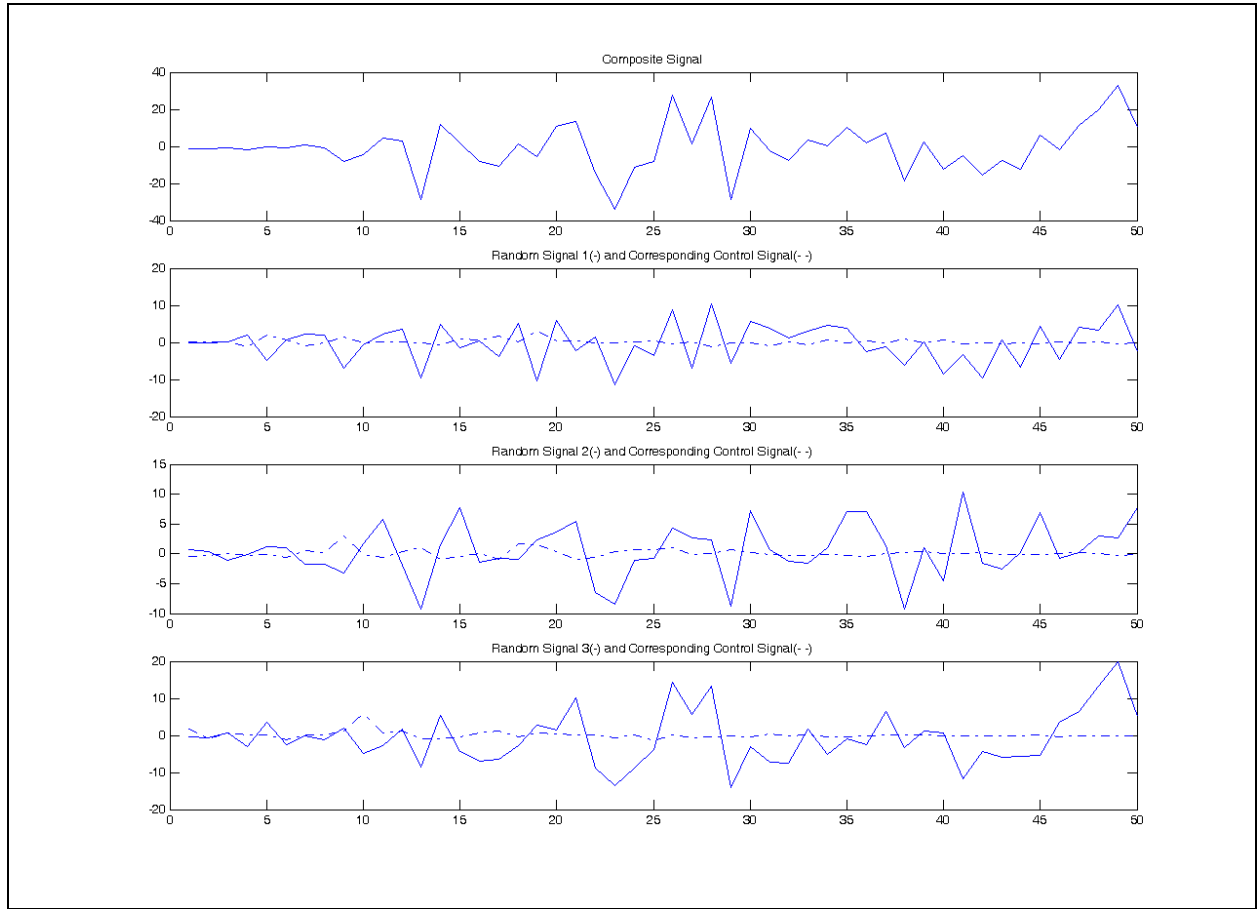
times another item becomes apparent. Better results are achieved when signal shifts are kept low. If the shifts are very close to the length of the signal then very little of the component signal will be included in the composite vector. The example given in Figure 2.5 illustrates this point.



**Figure 2.5 – Illustration of Component Signals Being ‘Lost’**

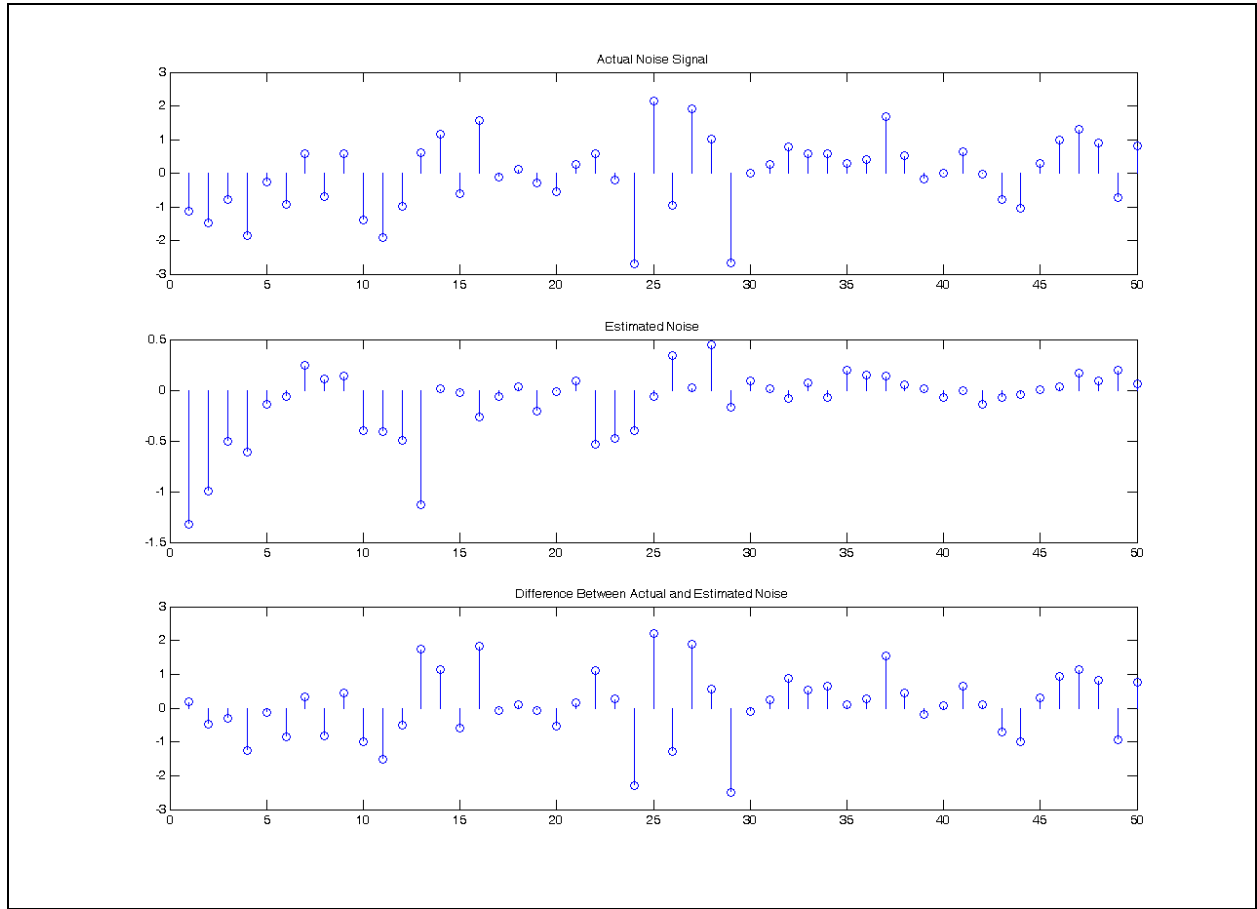
The control signals for random signals one and three have peaks at 17 and 3 respectively corresponding to an actual delay of 16 and 2. This is expected because since these signals start at 1 a delay of 16 would correspond to starting at data point 17. Unfortunately, the second control signal has no definitive peak. However the actual second control signal was shifted by 41, very close to the edge of the composite signal. As a result it is not surprising that this method failed to detect it.

Due to the relative success of the minimum energy approach when applied to completely uncorrelated signals it is possible to test how this method responds to noise. To test this a small noise signal will be added too the composite signal. The estimated noise signal will then be compared against the actual noise signal.



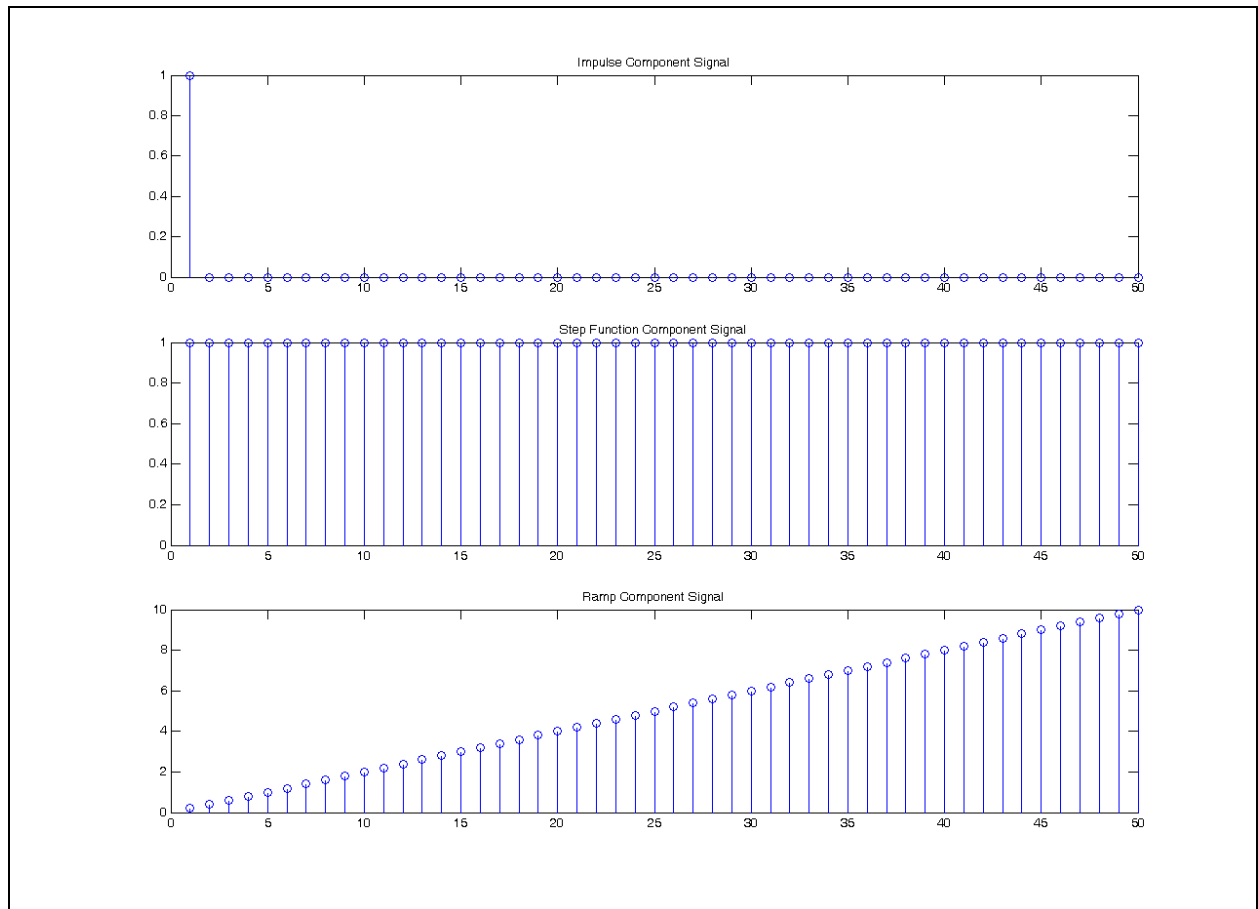
**Figure 2.6 – Effect of Noise on the Minimum Energy Approach**

While the large peaks do correspond with the correct delays, they are not quite as apparent as they were without noise. However, being as they were still identifiable, it would appear that this method has a slight amount of noise tolerance. Even so, when the actual noise signals are compared to the estimated noise signals, it is apparent that while this method has minor noise tolerance, it cannot be used for noise rejection as the difference between the actual noise and the estimated noise contains roughly the same power as the actual noise signal.



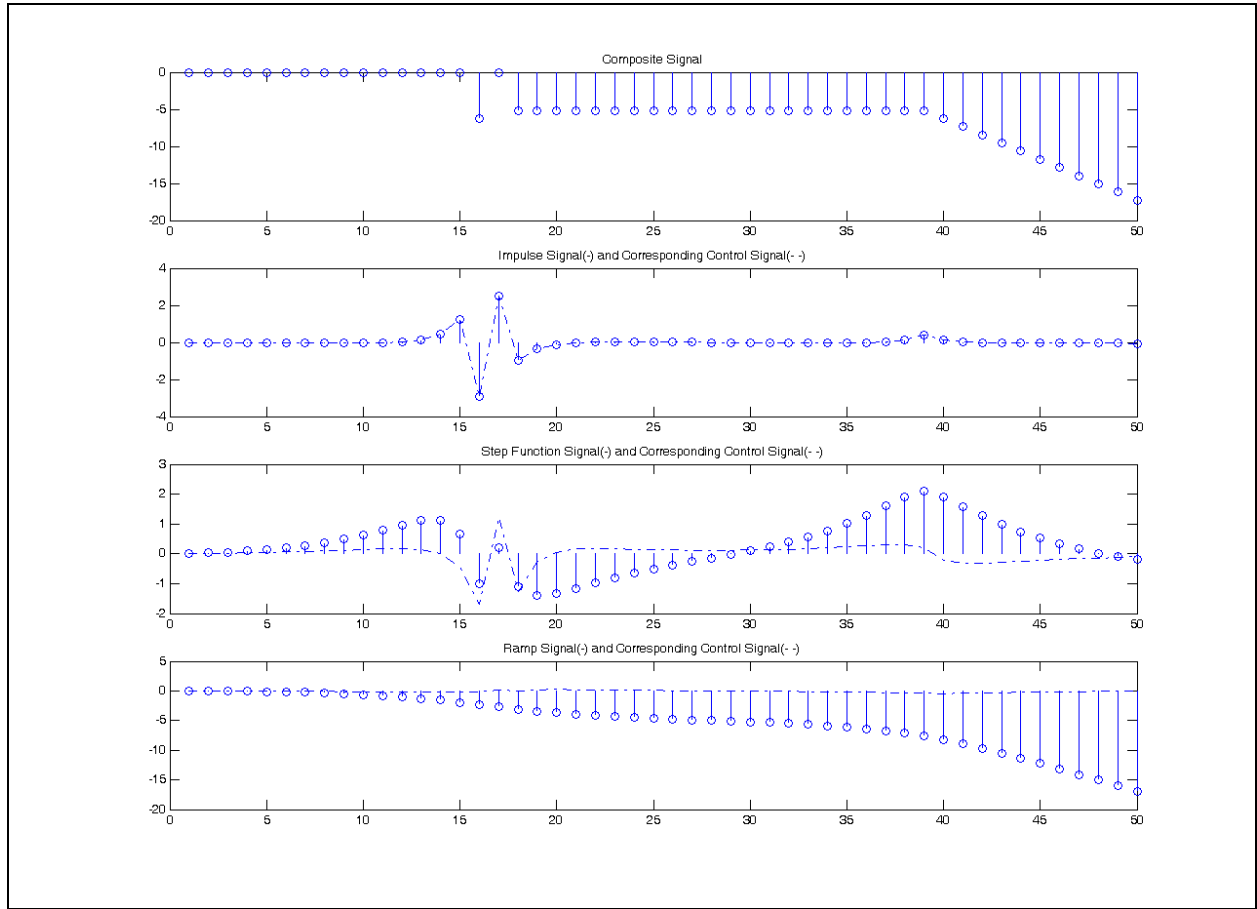
**Figure 2.7 – Attempt at Noise Removal in the Minimum Energy Approach**

The last set of components that this method will be tested on is the most important set for the scope of this thesis, that of an impulse, a step, and a ramp as given in Figure 2.8. This is considered the most important set of components for this thesis because each of the components are non-periodic, highly correlated with the other components vectors, and they combine to form composite signals that are simple enough to easily identify the intuitive solution.



**Figure 2.8 – Set of Non-periodic, Cross-Correlated Component Signals**

Applying the testing procedure for this set yields the following plots listed on Figure 2.9.



**Figure 2.9 – Results of the M.E.A to Non-periodic Cross-Correlated Component Signals**

In this example, where the intuitive solution is very obvious, the algorithm proves to be deficient. Yes the impulse was detected, but part of the step function was also classified as an impulse. The step component looks like a ramp while the ramp component looks like a quadratic.

## 2.4 Conclusions About the Minimum Energy Approach

The minimum error multi-signal deconvolution approach to multi-signal deconstruction provided several interesting results. First, while this method is able with moderate success to

generate an intuitive solution to signals composed of component signals that are uncorrelated and have no cross correlation it cannot guarantee this in a general result. This is not to say that this approach is without merit, only that it fails to generate a convincing approximation of the intuitive solution. Some variation of this method that takes into consideration the cross correlation matrices of the set of component signals may yet yield the intuitive solution. Recall that when this method was applied to systems composed of identically independent Gaussian vectors with essentially no covariance, the method worked moderately well. Although further processing would still be necessary to generate an entirely intuitive solution, the most important points were found, that being the signal type and the delay. Once these quantities are known finding the corresponding magnitudes becomes a relatively simple task. This principle will be further explored in the next chapter.

Even with the limited success that minimum energy multi-signal deconvolution achieved in multi-signal deconstruction, it is apparent that it is by no means an adequate solution to this problem. While a similar approach might solve some of its shortcomings, in its current form a more reliable solution is required.



### 3.0 Minimum Complexity Multi Signal Deconvolution

#### 3.1 Introduction

Because of the obvious shortcomings in the minimum energy approach a more robust and intuitively accurate method is required. For this method, instead of concentrating on a solution that relies on minimum energy, a method based on minimum complexity is proposed. This method has two requirements. The first is to make the error between the estimated composite signal and the actual composite signal as small as possible. The other criteria is to find the simplest possible explanation for the given composite signal. So while a step of 1 at every data point or an ever-increasing train of impulses could explain a ramp of slope 1, the simplest explanation is that it simply is a single ramp of slope 1. This makes much more intuitive sense than the minimum energy solution although it proves to be much more difficult to calculate.

#### 3.2 Derivation

##### 3.2.1 Characterize Data Signal

The first step in solving this problem is to model the composite signal in such a way as to allow the problem to be solved for minimum complexity. To do this the problem will again be set up as a summation. However instead of a summing over the number of component signal types, it will be summed over the number of component signals present in the composite signal. Here,  $y_k$  is composed of a sum of  $n$  component signals. Each component signal is a signal from the set of component signals  $c$  multiplied by a scalar  $a$  and shifted by a number of data points  $b$ .

$$y_k = \sum_{i=1}^n a_i s_{(i)k-b} \Big|_{k=b_i}^{bi+N-1} \quad (3.2.1)$$

In order to generate the proper solution to this problem the difference between the estimated composite signal and the actual composite signal must be minimized in addition to the number of component signals. The following is a possible cost function for this problem.

$$C(n, a, b, s) = \left| \tilde{y}_k - \sum_{i=1}^n a_i s_{(i)k-b} \right|_{k=b_i}^{bi+N-1}^2 \quad (3.2.2)$$

Although this does make use of a minimum energy type cost function it is not considered minimum energy. This is because the function is to be minimized first with respect to n. This is to say that this problem seeks to find solution that will find the smallest n that generates the minimum cost. Each n will have a corresponding set of vectors a and b that are a minimum energy solution to the problem. Increasing n should decrease the cost function. However at some value of n it is assumed that further increases of n will have a negligible effect on the cost function. This extremely complicated problem does not have an immediately apparent solution, and as a result an alternative method is required to generate a minimum complexity solution.

### 3.2.2 Numeric Method Proposed

Typically when a closed form solution to a problem either cannot be found or is simply too difficult to find, one looks for some sort of numerical method to solve the problem. In this case a numeric method is required to accurately find the minimum number of component signals as well as the magnitude, delay, and type of component signals used to form a composite signal. The approximate composite signal must also be within some error threshold of the original composite signal determined by the additive noise power. To begin, a brute force approach will be used, followed by a more sophisticated approximation.

### 3.2.3 Assume Number, Types and Delays of Component Signals are Known

As a starting point for this algorithm we will assume a very simple circumstance. Suppose we know the number of component signals, the types of component signals and the corresponding delay of each component signal. This greatly simplified problem becomes simple to solve. The problem has been reduced to one of the following form, given a set of n signals find a corresponding set of scalar multiples such that the summation of the products of each signal with its corresponding scalar provides the best possible representation of the signal. Because the best possible representation is usually taken to mean the least squared error the problem can be set up as follows.

$$\varepsilon^2 = \left| \tilde{y}(k) - \left( \sum_{i=1}^n a_i s_i(k) \right) \right|^2 \quad (3.2.3)$$

Fortunately a great deal of work has been done on this problem, most notably is the work done involving the projection theory [12]. In this case each s is assumed to be a basis function. Applying the projection theory yields equation (3.2.4) where  $\langle s_1, s_1 \rangle$  is defined as  $s_1' s_1$ .

$$\begin{bmatrix} \langle s_1, s_1 \rangle & \langle s_1, s_2 \rangle & \langle s_1, s_3 \rangle & \cdots & \langle s_1, s_n \rangle \\ \langle s_2, s_1 \rangle & \langle s_2, s_2 \rangle & \langle s_2, s_3 \rangle & \cdots & \langle s_2, s_n \rangle \\ \langle s_3, s_1 \rangle & \langle s_3, s_2 \rangle & \langle s_3, s_3 \rangle & \cdots & \langle s_3, s_n \rangle \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \langle s_n, s_1 \rangle & \langle s_n, s_2 \rangle & \langle s_n, s_3 \rangle & \cdots & \langle s_n, s_n \rangle \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} \langle y, s_1 \rangle \\ \langle y, s_2 \rangle \\ \langle y, s_3 \rangle \\ \vdots \\ \langle y, s_n \rangle \end{bmatrix} \quad (3.2.4)$$

This equation is composed of three matrices. A cross correlation matrix and a vector of unknown magnitudes sit on the left of the equation. On the right sits the projection of the signal in question to each of given signals s. To solve this equation we must invert the cross correlation matrix, however this matrix is not guaranteed to be non-singular. As a result another technique is

proposed. Instead of forcing the covariance matrix to have an inverse the well-known approximation given in (3.2.5a) will be used.

$$\text{cov}_s^{-1} \approx \text{cov}_s (\text{cov}_s \text{cov}_s')^{-1} \quad (3.2.5a)$$

This is guaranteed to have an inverse because the covariance matrix is guaranteed to be of full rank, as a primitive signal can not be a linear combination of other primitive signals. With an approximation for the covariance matrix given in (3.2.5a), the magnitudes can be found using the equation given in (3.2.5b).

$$\begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ \overline{a_n} \end{bmatrix} = \begin{bmatrix} \langle s_1, s_1 \rangle & \langle s_1, s_2 \rangle & \langle s_1, s_3 \rangle & \cdots & \langle s_1, s_n \rangle \\ \langle s_2, s_1 \rangle & \langle s_2, s_2 \rangle & \langle s_2, s_3 \rangle & \cdots & \langle s_2, s_n \rangle \\ \langle s_3, s_1 \rangle & \langle s_3, s_2 \rangle & \langle s_3, s_3 \rangle & \cdots & \langle s_3, s_n \rangle \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \langle s_n, s_1 \rangle & \langle s_n, s_2 \rangle & \langle s_n, s_3 \rangle & \cdots & \langle s_n, s_n \rangle \end{bmatrix}^{-1} \begin{bmatrix} \langle y, s_1 \rangle \\ \langle y, s_2 \rangle \\ \langle y, s_3 \rangle \\ \vdots \\ \langle y, s_n \rangle \end{bmatrix} \quad (3.2.5b)$$

### 3.2.4 Finding the Signal Types and Delays

Having solved the simplest possible situation we must now move on to a more difficult problem, that being a situation where only the number of primitives contained in the signal is known. The type of signals used to create the composite vector and their corresponding magnitude and delay factors remain unknown. To solve this problem an assumption will be made about the intuitive solution. That being given a number of unique possible representations of the given data signal, each having the same number of signals with magnitudes calculated such that they form the best possible fit for the given signal types and delays, the best solution will be the one with the smallest error between the approximate signal and the given signal. This can be

further expanded to state that representations that have an equal amount of error between the actual signal and the approximate signal are equivalently intuitive.

With this assumption a method can be constructed to find the most intuitive possible solution. Given  $n$  component signals of length  $N$  selected from a set of  $k$  component signal types, it is possible to scan through all possible combinations of signal types and delays. For each combination a best fit can be made and the error between the approximated signal and the actual composite signal can be found. The final solution is the set that has the smallest error, or the set that forms the best explanation for the given composite signal.

An algorithm is needed to generate all possible combinations of signal types and shifts. Fortunately Matlab has such a function. The function 'nchoosek' can be used to generate a vector of all possible combinations of selecting  $n$  signals from a vector containing all possible signals and shifts.

### **3.2.5 Determining the Minimum Number of Component Signals**

Now all that is left to do is to find a method for determining the number of component signals present in a given composite signal. One last assumption about the intuitive solution is required to complete this numerical method. The intuitive solution is assumed to be the least complicated solution. This means that solutions containing fewer component signals are preferred over solutions that require more components. Applying this last assumption, an intuitive solution can be defined by the solution with the least number of component signals that produces an estimated signal that comes within a given error margin to the original composite signal.

The algorithm then begins, starting with the assumption that the number of component signals is zero. It then determines if the estimated signal, in this case a vector of zeros, is within a given tolerance to the actual composite signal. If it is within the given tolerance the algorithm outputs the results. If the error is greater than the tolerance, the algorithm then considers the case

when there is one component signal. It searches through all possible signal types and delays. For each combination, it calculates the appropriate magnitudes using a least squared error algorithm. For each combination it then calculates the error between the estimated signal and the actual signal. It then must compare the smallest error to the tolerance given. If the smallest possible error for a single component signal is within the given tolerance the algorithm outputs the results. If the smallest possible error is not within the given tolerance, the algorithm increments the number of signal components to two and repeats the aforementioned process. It continues to increment the number of signal components until a solution is found that is within the accepted tolerance.

### **3.2.6 Determining the Proper Threshold**

One question that has not yet been answered is how to determine the tolerance between the signal comprised of the estimated component signals and the given composite signal. In a perfect world with no noise or numerical inaccuracies the tolerance for the minimum complexity algorithm could simply be set at zero. Unfortunately however, both occur and therefore must be taken into account. The problem that arises is that if the tolerance is set too low, the best solution could be overlooked in favor of a solution with a greater than necessary number of component signals. If the tolerance is set too high, a solution with too few signal components could be selected. As a result an intelligent choice of tolerance should be used.

Although both noise and numerical inaccuracies can occur and the net tolerance must take both factors into consideration the problem can be divided into two separate cases because noise is generally much greater than errors caused by numerical inaccuracies. The two cases can be defined as systems with numeric inaccuracies and systems with additive noise.

In the case of systems with numerical inaccuracies, choosing a tolerance is relatively simple. Setting the tolerance to a small constant value such as .001 will work quite well for

typical systems. This is because this number is small enough to avoid losing signals and large enough to correct numerical inaccuracies. However, systems with either very large or very small typical values should choose a tolerance based on the expected numerical inaccuracy. The approach used to determine the tolerance based on noise could be used in this case.

Choosing a tolerance in the general case relies heavily on the art of guesstimation. However it can be insightful to consider the following case. Suppose the system has been exposed to additive Gaussian noise of a known power  $\sigma$ . It is possible to find the cumulative density function for the noise power over a given segment of the signal. From the cumulative distribution function, or CDF, it will be possible to find not only an intelligent tolerance value, but also a measure of the validity of the results returned by the algorithm. In order to find the cumulative density function for the sum of a set of squared independent noise values one must start by examining the probability density function for a given point of noise [14].

$$p(x_i) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x_i^2}{2\sigma^2}} \quad (3.2.6)$$

To simplify the calculation of the desired CDF a new variable  $y$  will be defined to represent the power of one sample of noise.

$$y = x_i^2 \quad (3.2.7)$$

Because  $y$  is not linearly related to  $x$  the simplest method to find the probability density function for this new function  $y$ , makes use of the characteristic function.

$$\psi(y) = E\{e^{jyu}\} = E\{e^{jx_i^2 u}\} \quad (3.2.8)$$

Because the expected value of a function is defined as the integral from negative infinity to infinity of a function multiplied by its probability density function the characteristic function of  $y$  can be written as follows.

$$\psi(y) = \int_{-\infty}^{\infty} e^{jx_i^2 u} \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x_i^2}{2\sigma^2}} dx \quad (3.2.9)$$

Now taking into account that  $dx = dy \cdot (4y)^{-0.5}$ ,  $y$  can be substituted in to the expression.

$$\psi(y) = \int_0^{\infty} e^{jy u} \frac{1}{\sigma\sqrt{2\pi y}} e^{\frac{-y}{2\sigma^2}} dy \quad (3.2.10)$$

This is exactly in the form of a characteristic function. Therefore,  $y$  or the value of the error squared caused by one noise sample of a noise series having power of sigma has the following probability density function.

$$p(y) = \frac{1}{\sigma\sqrt{2\pi y}} e^{\frac{-y}{2\sigma^2}} \quad (3.2.11)$$

Having found the probability density function of the power of a single series of noise we must now find the probability density function for the power contained in an  $N$  point series. To do this a second variable  $y'$  will be defined as the  $N$  point sum of  $y$ .

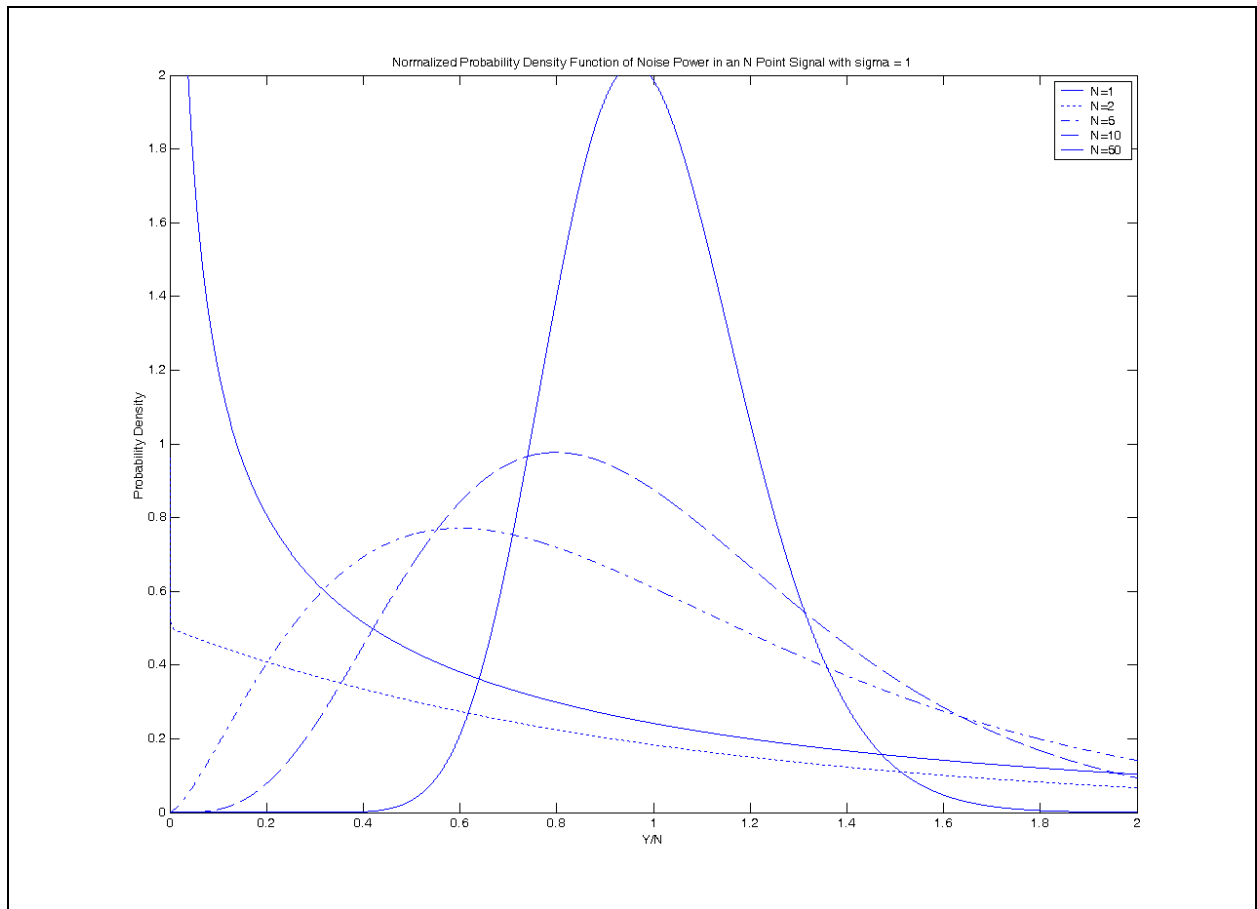
$$y' = \sum_{i=0}^N y_i = \sum_{i=0}^N x_i^2 \quad (3.2.12)$$

When summing independent variables, the probability density function of the resulting summation can be calculated by convolving each of the probability density functions of the variables being summed. As a result, the probability density function of  $y'$  can be calculated by the net convolution of  $N$  pdf's of  $y$ . This is a very difficult expression to use. To proceed to a meaningful expression it will be necessary to revert back to the characteristic functions. This is a more useful tactic because convolution of two probability density functions is equivalent to multiplying their characteristic functions.



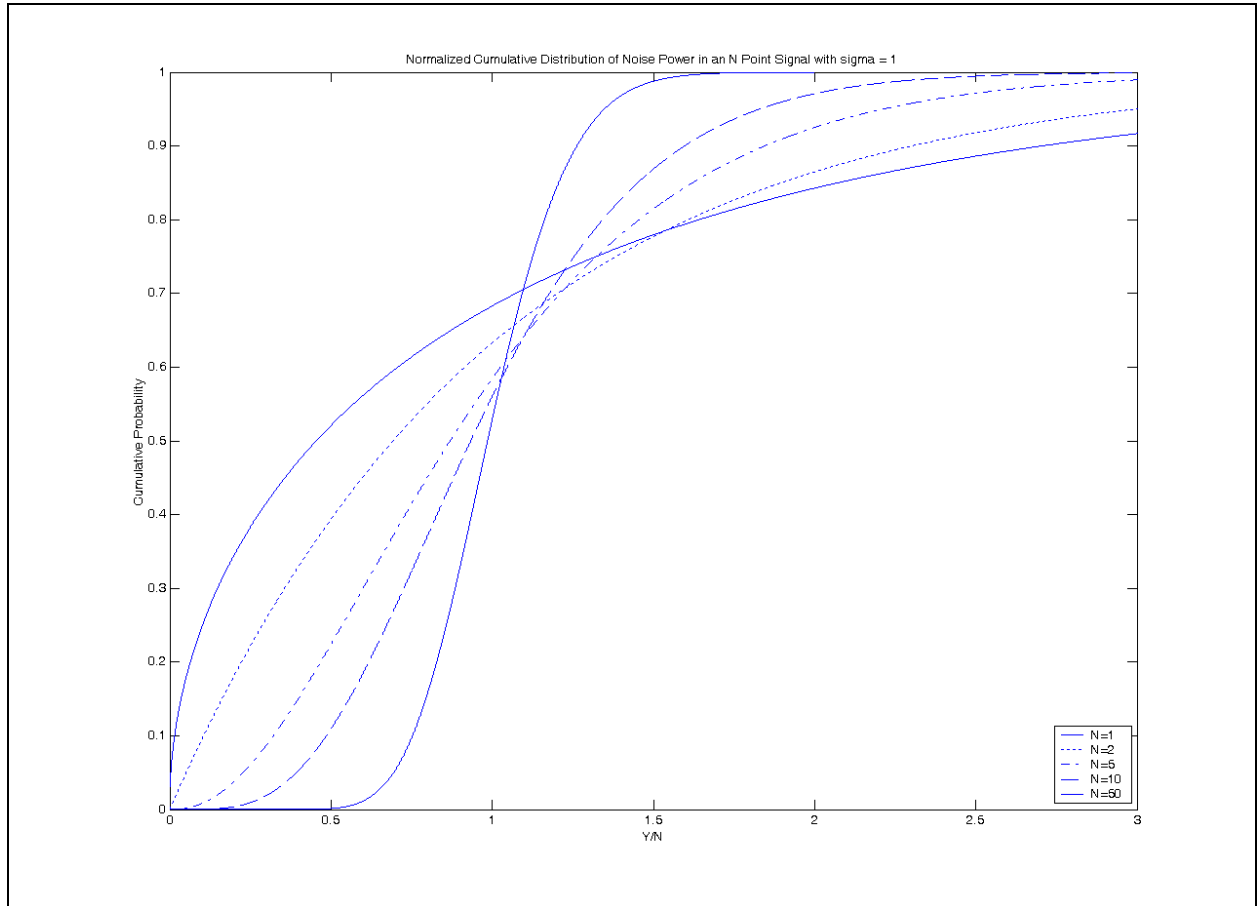
$$\psi(y') = \prod_{i=0}^{N-1} \int_0^{\infty} e^{jy'u} \frac{1}{\sigma\sqrt{2\pi y}} e^{\frac{-y_i}{2\sigma^2}} dy \quad (3.2.13)$$

Proceeding onward from this point along an analytical approach becomes extremely difficult. To ease the complexity of the calculating the cumulative probability distribution the problem will be solved using a discrete approximation. It is possible to find the characteristic equation of  $y$  by taking the FFT of  $y$ 's probability density function. Taking the IFFT of this result generates an approximation for the probability density function of  $y'$ . This can then be integrated to generate an approximate cumulative distribution function. To gain some understanding from this distribution Figures 3.1 and 3.2 show the pdf and the cdf of the noise power of various length noise signals normalized by dividing  $y$  by  $N$ . This allows us to view a distribution of the average noise power.



**Figure 3.1 – PDF's of the Total Energy in White Noise Signals of Various Lengths**

As one would expect, the noise distribution becomes more Gaussian and also has a smaller variance the higher N becomes.



**Figure 3.2 - CDF's of the Total Energy in White Noise Signals of Various Lengths**

Looking at the CDF it is possible to see the importance of a correctly chosen threshold. Considering Figure 3.2, if one wanted to set the threshold at the point that would ensure a 90 percent chance that the noise power was less than a certain value one must take into account the amount of data considered. For data signals of length five more than twice the tolerance of a signal of length 50 is required. All these considerations combined with the fact that the actual noise power is not known with certainty make it very difficult calculate the proper threshold. Fortunately, through testing a number of different signals, a set of values that seem to work well

are thresholds from  $1.1*N*\sigma$  to  $2*N*\sigma$ . Thresholds in this region seem to be a good compromise between forcing too many component signals and allowing too few signals to be returned.

One method of using the thresholds not considered in this paper is to use an upper and a lower threshold instead of just an upper bound on the noise. This would reject signals solutions that don't have enough error. This is not considered however, because it would require the least squares estimate to be much more complicated, consuming more computing resources.

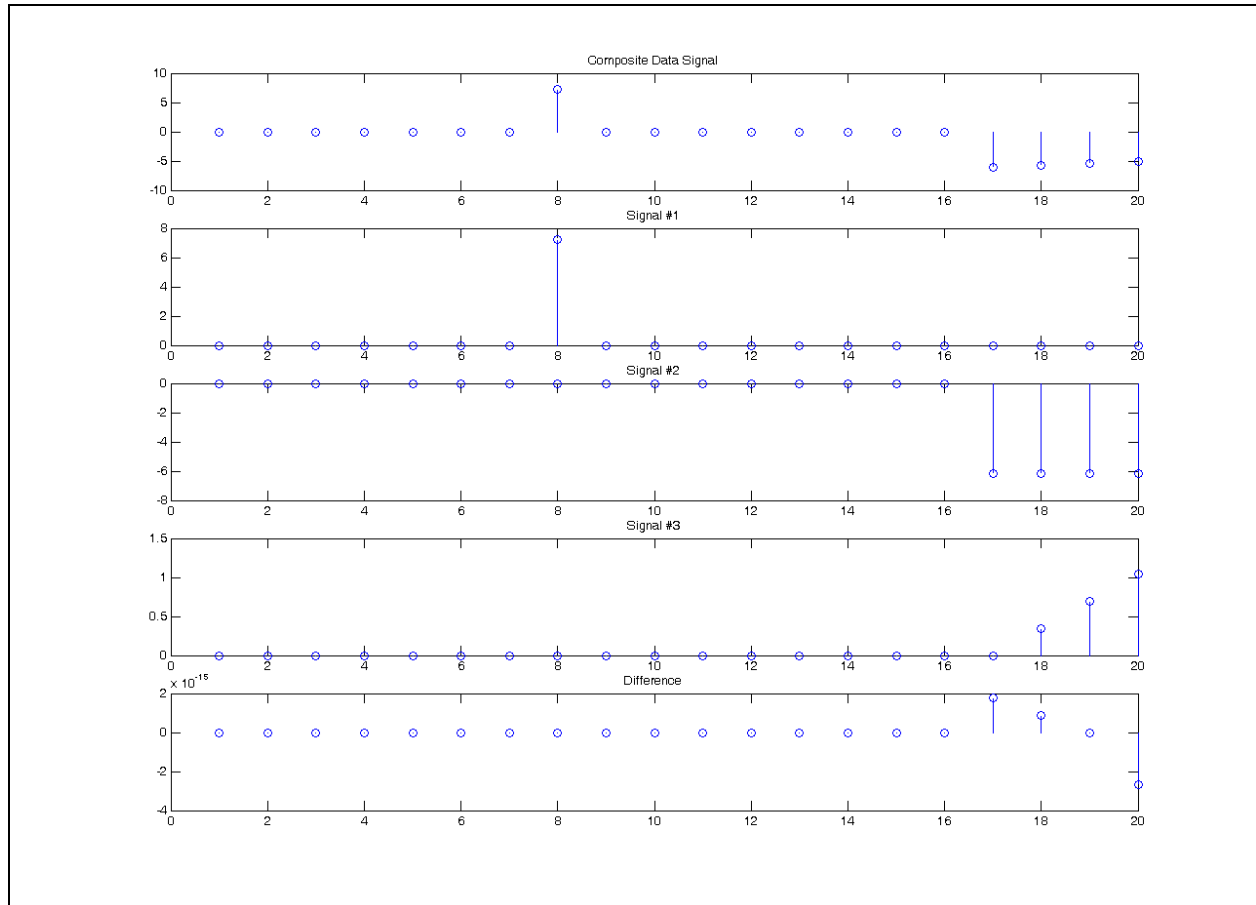
### **3.2.7 Signal and Magnitude Constraints**

One concern of some systems is that certain answers might not make sense. For example a system might have constraints where all magnitudes are greater than a certain value. To compensate for this, additional criterion can be added to the algorithm to support such special systems. Due to the nature of such systems solutions to these nonlinear problems require nonlinear constraints to be placed on the least squares approximation algorithm. This is not covered in detail in this paper, it is only mentioned to illustrate the flexibility of this algorithm.

## **3.3 Testing**

Now that an algorithm has been devised, it must be tested to determine if the minimum complexity approach actually provides the intuitive solution. To test this the final example from the minimum energy approach will be considered. Only this system will be contemplated because of two factors. The first reason is that a composite signal made up only of ramps, steps, and impulses is a very easy to decompose by eye into the intuitive solution. Second, as shown by the minimum energy approach, due to numerous factors this is a difficult signal to decompose into the intuitive solution. When the minimum energy approach was applied on this system, it completely failed to generate any relevant results.

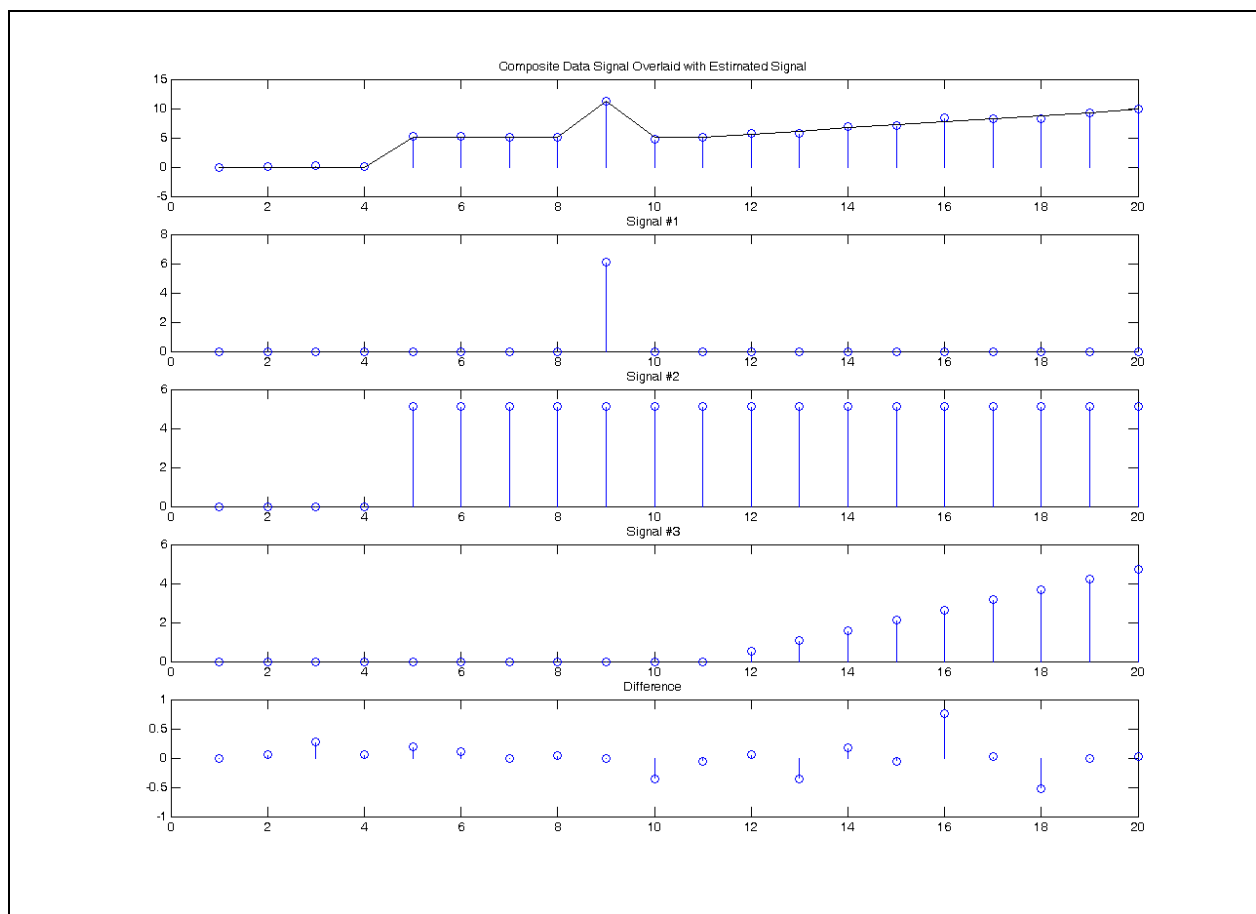
To begin testing this procedure, a 20-point composite signal, as shown in Figure 3.3, will be used containing a single impulse, step and ramp.



**Figure 3.3 – Results of the Minimum Complexity Algorithm to a Noise Free Signal**

From the first test this method appears very promising. The algorithm had no problem finding each of the three signals we would expect from the intuitive solution. Looking at the

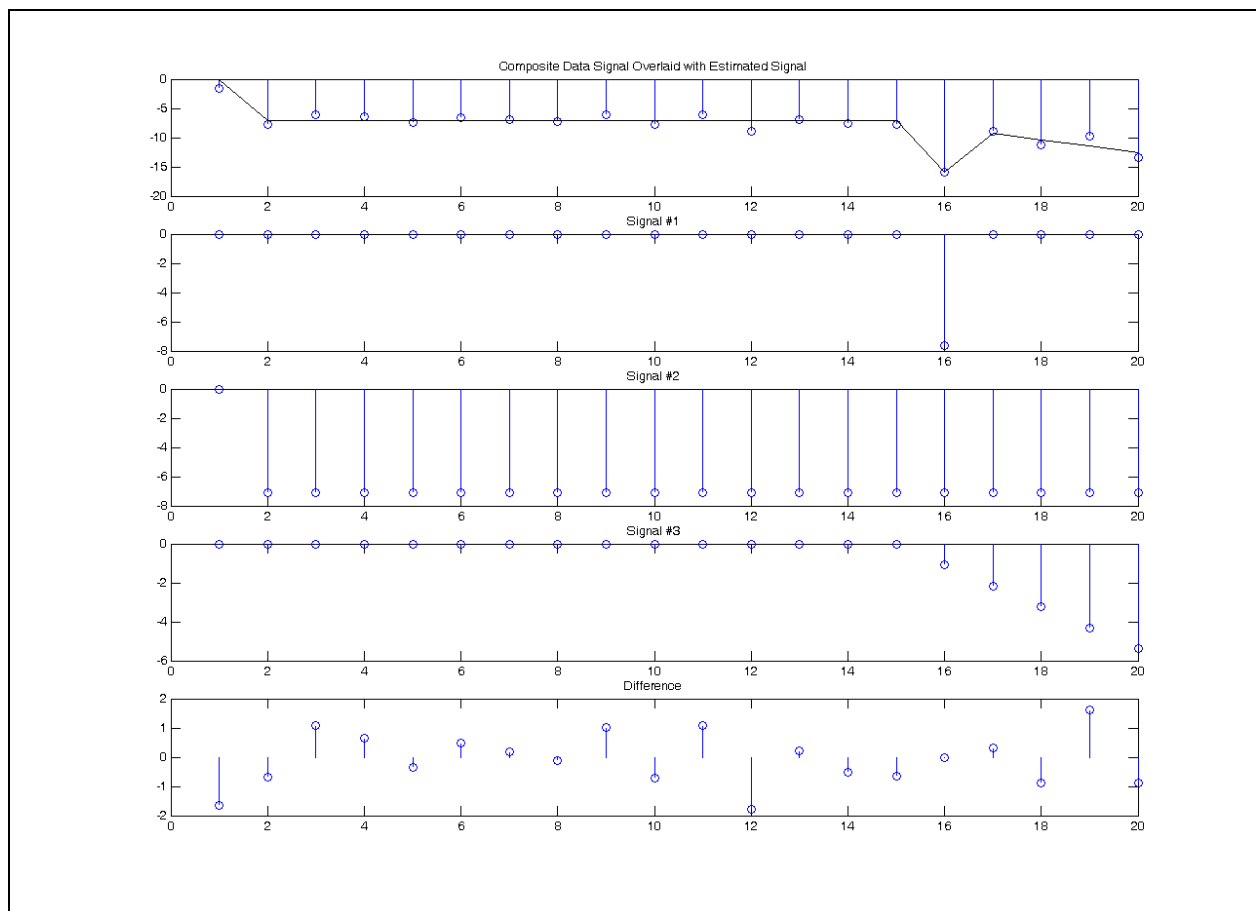
error signal it is clear that the only difference between the approximated signal and the actual composite signal was a small amount of numerical inaccuracy. This method apparently does quite well on very clean, noise free signals. Repeating the test with a small amount of Gaussian white noise, of power .1 provides the following output shown in Figure 3.4.



**Figure 3.4 – Results of the M.C.A. a Signal with Small Additive Noise**

Again this algorithm appears to have obtained the correct intuitive solution. The noise has been successfully removed from the approximated signal. Each of the 3 component signals have been correctly identified and are of approximately the correct magnitude, some error in magnitude is expected, for example it is impossible to know the actual value of an impulse since it is colored with noise.

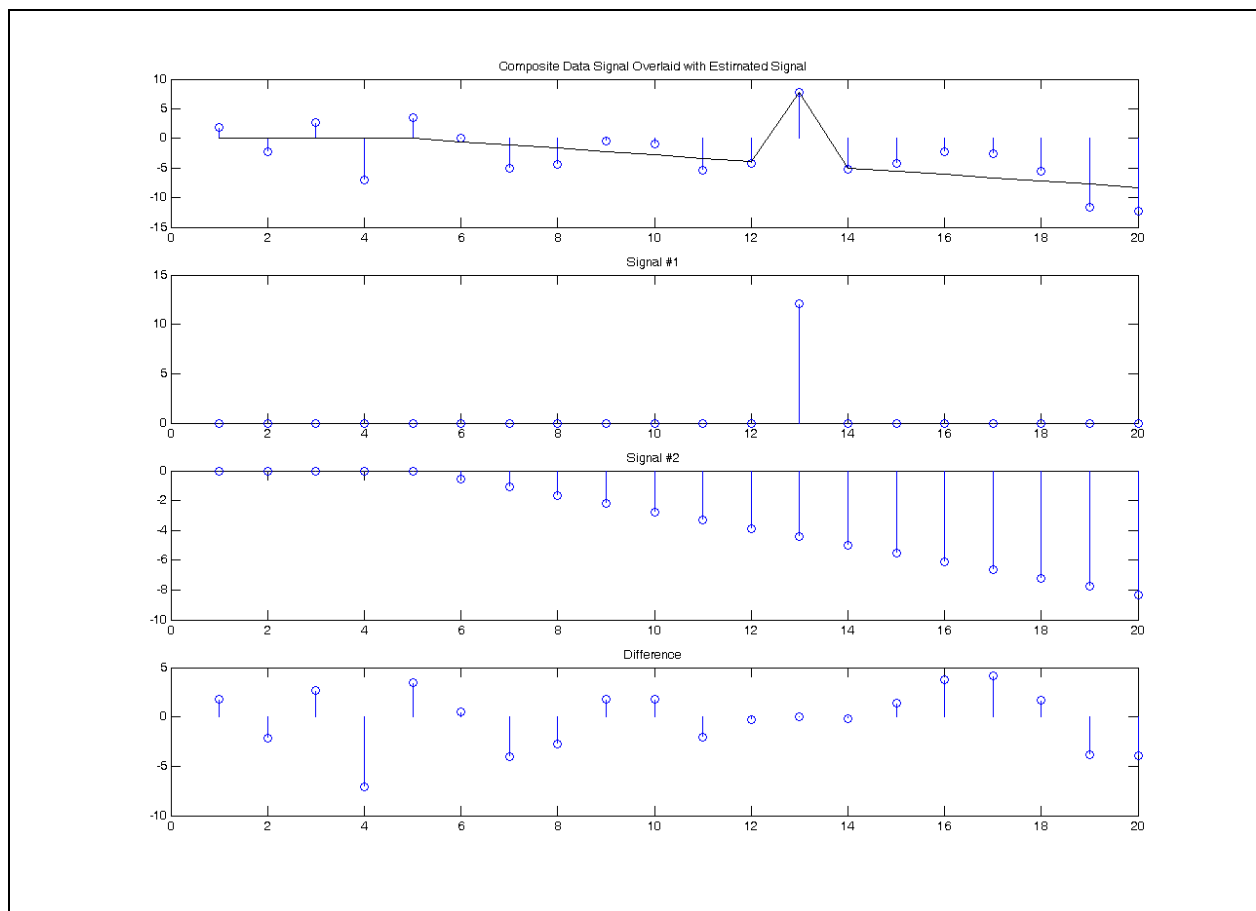
To test the system further the noise power was extended to 1.



**Figure 3.5 – Results of the M.C.A. to a Signal with Moderate Additive Noise**

Again the system appears to agree with our notion of an intuitive solution. Although some simulations with these parameters failed to find each of the three signals, typically accurate results were returned. This does not mean that on those occasions where the algorithm failed to find all three component signals that the algorithm returned an unintuitive solution, rather that the error added to the created signal skewed the intuitive solution into a simpler solution.

As is shown in Figure 3.6, it was only when the additive noise approached a power of 10 that algorithm was unable to consistently find the three component signals.



**Figure 3.6 - Results of the M.C.A. to a Signal with Large Additive Noise**



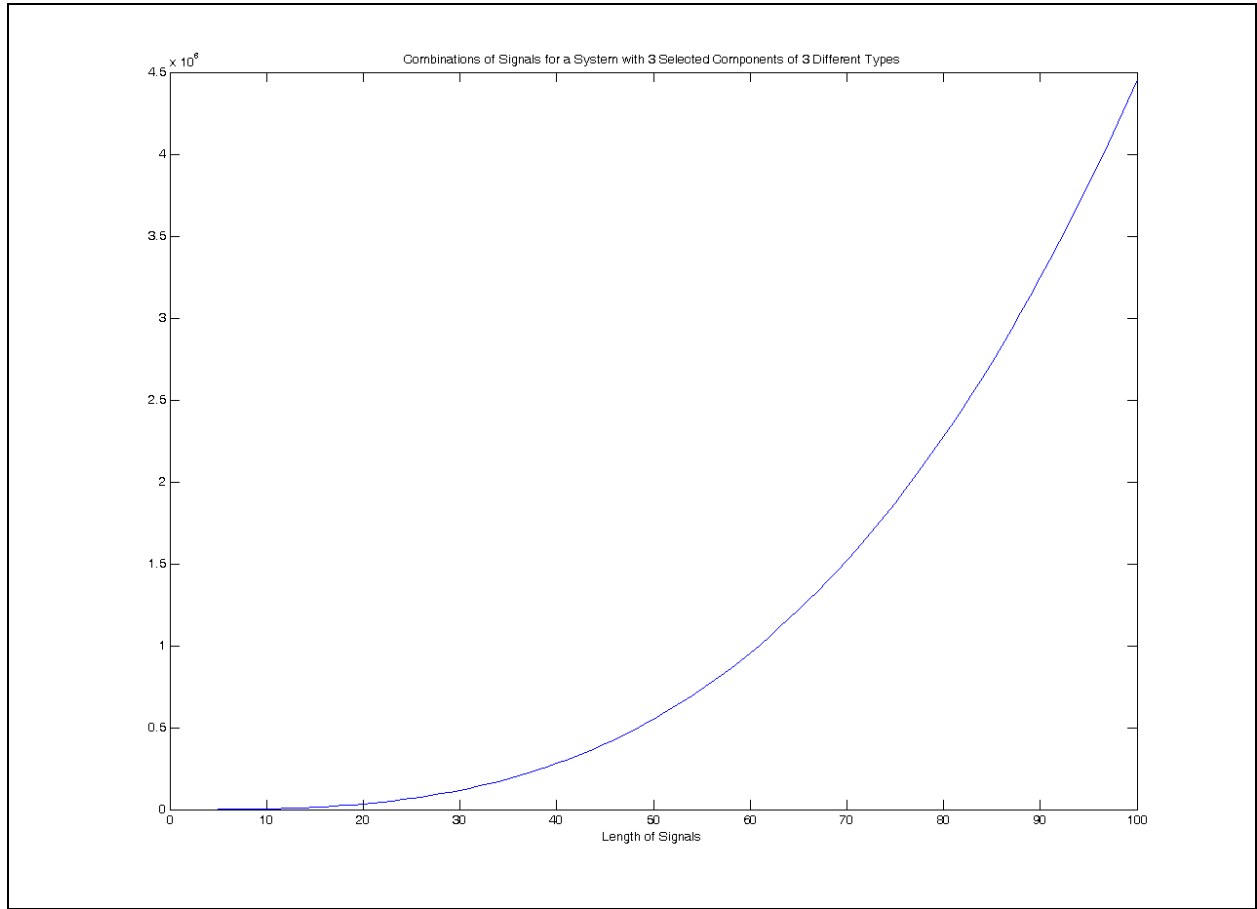
In the case of Figure 3.6, the algorithm was only able to detect a ramp and an impulse instead of a ramp, a step, and an impulse. However, it is difficult to make a case for a step being in the intuitive solution, if the noise happens to obliterate the step function then there is no step function in the intuitive solution. Therefore, it is possible to say that this algorithm is quite accurate in terms of finding the intuitive solution.

### 3.4 Performance of Algorithm

While this method does present a solution to a difficult problem it comes at a cost. With  $n$  signals chosen from a set of  $k$  types each with  $N$  possible delays a very large number of iterations can be required. Using the binomial theorem the following expression can be derived determining the number of iterations required.

$$\#Iterations = \binom{N \cdot k}{n} = \frac{(N \cdot k)!}{n!(N \cdot k - n)!} \quad (3.4.1)$$

As the following plot illustrates, signals with only moderately large lengths can generate an immense number of signal combinations.



**Figure 3.7 – M.C.A. Iterations vs Signal Length**

Even more than additional signal length, considering additional signals can generate even more immense computational loads.

Iterations Required for Different Situations								
	<i>Length of Composite Signal</i>							
	5	10	15	20	25	30	35	40
<i>Number of Component Signals</i>	1	15	30	45	60	75	90	105
	2	105	435	990	1770	2775	4005	5460
	3	455	4060	14190	34220	67525	117480	187460
	4	1365	27405	148995	487635	1215450	2555190	4780230

**Table 3.1 – M.C.A. Iterations for Selected Number of Component Signals and Data Lengths**

## **4.0 Fast Minimum Complexity Algorithm**

### **4.1 Introduction**

To solve this last problem, that of the excessive computational time needed for large composite signals composed of many component signals, a windowing solution is proposed. It is hoped that for a given system, a windowing method can be found such that the total number of iterations are dramatically reduced without sacrificing adherence to the intuitive solution.

### **4.2 Proposed Solution**

Without defining the individual quantities yet, a basic outline for this algorithm consists of a loop that slides a window along the composite signal containing three basic parts. Due to the nature of the loop it is easier to begin to explain the fast algorithm from the second stage. In the second stage of the loop the partial composite signal is passed on to the standard minimum complexity algorithm. Next, the resulting component signals that have been detected within the region of the data that will be incremented out of the sampling window are stored. The window is incremented and the loop resets. Looking back to the first step, the stored component signals are used to construct a predicted sample over the current window. This quantity is then subtracted from the windowed composite signal. The result is the signal sent into the minimum complexity algorithm.

One of the great advantages of this method when combined with a system composed of only ramps, steps, and impulses is that the estimated signal over any given range can be exactly represented by only two quantities, an initial value and a slope. The initial value removes constant offset caused by ramps and steps while the slope counter compensates for the ramps. This greatly simplifies the algorithms used to generate this estimated signal.

## 4.3 Algorithm Parameters

### 4.3.1 Window Size

Selecting a proper window size is of great importance in this algorithm. Select too large of a window and the problems that applied in the original algorithm will again appear. Select too small of a window and there might not be enough data points for the algorithm accurately recreate the actual signal minus the additive noise. Using the noise analysis done on the original problem, it will be possible to select a valid window.

While the actual best size for a window depends largely on the given signal measured it is possible to form a few general rules. It is generally good to limit the window to a size where one can be relatively sure that of only one or two component signals are present at a time. Occasionally three component signals are acceptable, but with current computing speeds, this can create problems with anything more than around 20 data points. The window size should be somewhere in the region of 5 to around 150 points depending on the maximum number of signals expected in one window. For signals with three component signals, the window should be limited to around 10 or 20 points. If only two components are expected 10 to 50 data points can be used. Anything above this results in far too many iterations for realistic signals for current computing capabilities. The last item to consider when choosing a window is the noise power. To maintain the same relative accuracy, composite signals with large noise signals require more data points than composite signals of lower noise. I have found that given the data that I have worked with, (three component signals stretched over 100 or so samples), setting the noise tolerance to  $1.3 \cdot N \cdot \sigma$ , yields good results when the window is set in the region from 10 to 20 samples, depending on the noise. A higher noise level will require a larger window.

### 4.3.2 Window Increment

The reason only the signals whose delays are less than the window increment are stored is that it is more difficult to accurately calculate signals with fewer data points. For example, if a spike was found at the very last data point of a window, it would be impossible to tell if that peak corresponded to an impulse, a step, or a ramp. When noise is considered it becomes even more important to consider many data points. Only by considering many data points can the error between the estimated and actual composite signals be reduced. Therefore to eliminate errors, the smallest number of signals possible will be stored. A good rule is that the increment should never be more than half the window size. This is because successive windows should overlap, creating a minimum data support of the window length minus the window step size.

## 4.4 Performance Enhancement of Fast Algorithm

Due to the immense computational load that the original minimum complexity algorithm required, it is important to illustrate how much of a reduction that this method presents. To determine this the number of iterations for the fast minimum complexity algorithm must be considered. For an  $N$  point signal with a window length  $L$  and a window increment of  $I$  the number of iterations of the outer loop are as follows.

$$\begin{aligned}\#Iterations_{outer} &= \frac{N - (W - I)}{I} \\ \#Iterations_{outer} &= \frac{N - W}{I} + 1\end{aligned}\tag{4.4.1}$$

The number of iterations in the inner loop, which is the number of iterations required by the original minimum complexity algorithm over the window length is much more difficult to determine. This is because the number of iterations depends very highly on the grouping of component signals. A much larger number of iterations are required if the component signals are

grouped such that one window has 5 component signals while all of the other windows have none rather than all of the windows having one or two iterations. This is due to the nonlinear relationship between number of component signals and number of iterations. To solve this problem a probable upper bound on the number of iterations will be used instead of a probable number of iterations. A variable  $n$  will be defined as the highest number of component signals expected in one window length of data. With this estimate the probable maximum number of inner iterations over a window of length  $L$  and a component set of  $k$  signals can be written as follows.

$$\#Iterations_{inner} = \binom{L \cdot k}{n} = \frac{(L \cdot k)!}{n!(L \cdot k - n)!} \quad (4.4.2)$$

Combining the number of inner and outer iterations, the total number of iterations for the fast minimum complexity algorithm is much lower than the number of iterations of the original algorithm.

$$\begin{aligned} \#Iterations_{total} &= \#Iterations_{outer} \cdot \#Iterations_{inner} \\ \#Iterations_{total} &= \frac{(L \cdot k)!}{n!(L \cdot k - n)!} \left( \frac{N - W}{I} + 1 \right) \end{aligned} \quad (4.4.3)$$

To illustrate just how much of a difference this can make the following table compares the number of iterations required by both algorithms for different data lengths, and window sizes assuming one component signal every 10 samples and each window increment set at one half the window length.

Iterations Required for Different Situations								
Algorithm Type	Window	Length of Composite Signal						
		30	60	90	120	150	180	210
FMCA*	10	150	330	510	690	870	1050	1230
FMCA*	20	3540	8850	14160	19470	24780	30090	35400
FMCA*	30	117480	352440	587400	822360	1057320	1292280	1527240
OMCA**	NA	117480	43424719800	1.83649E+16	8.21885E+21	3.7953E+27	1.78791E+33	8.53978E+38

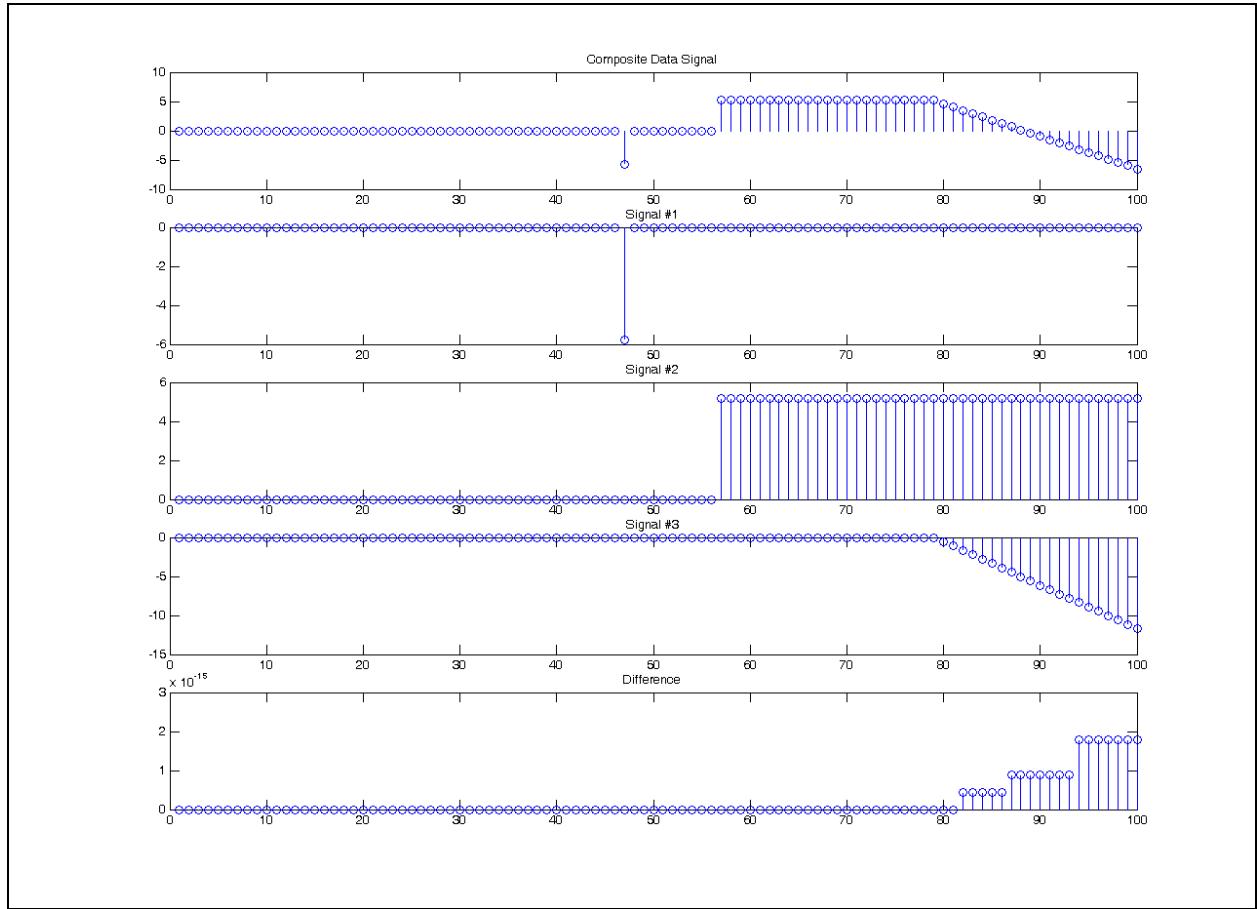
Table 4.1 – Comparison of Iterations for Several Data Lengths

The results are a very large increase in performance. In fact, a fast minimum complexity algorithm with a window of 30 operating on a 230 samples runs more than twenty five thousand times faster than the original minimum complexity algorithm running on 40 samples.

#### 4.5 Algorithm Performance

Implementing this algorithm on actual data signals yielded promising results. With a window of only 10 and a step size of 5 the algorithm was completed extremely fast, with very accurate results.

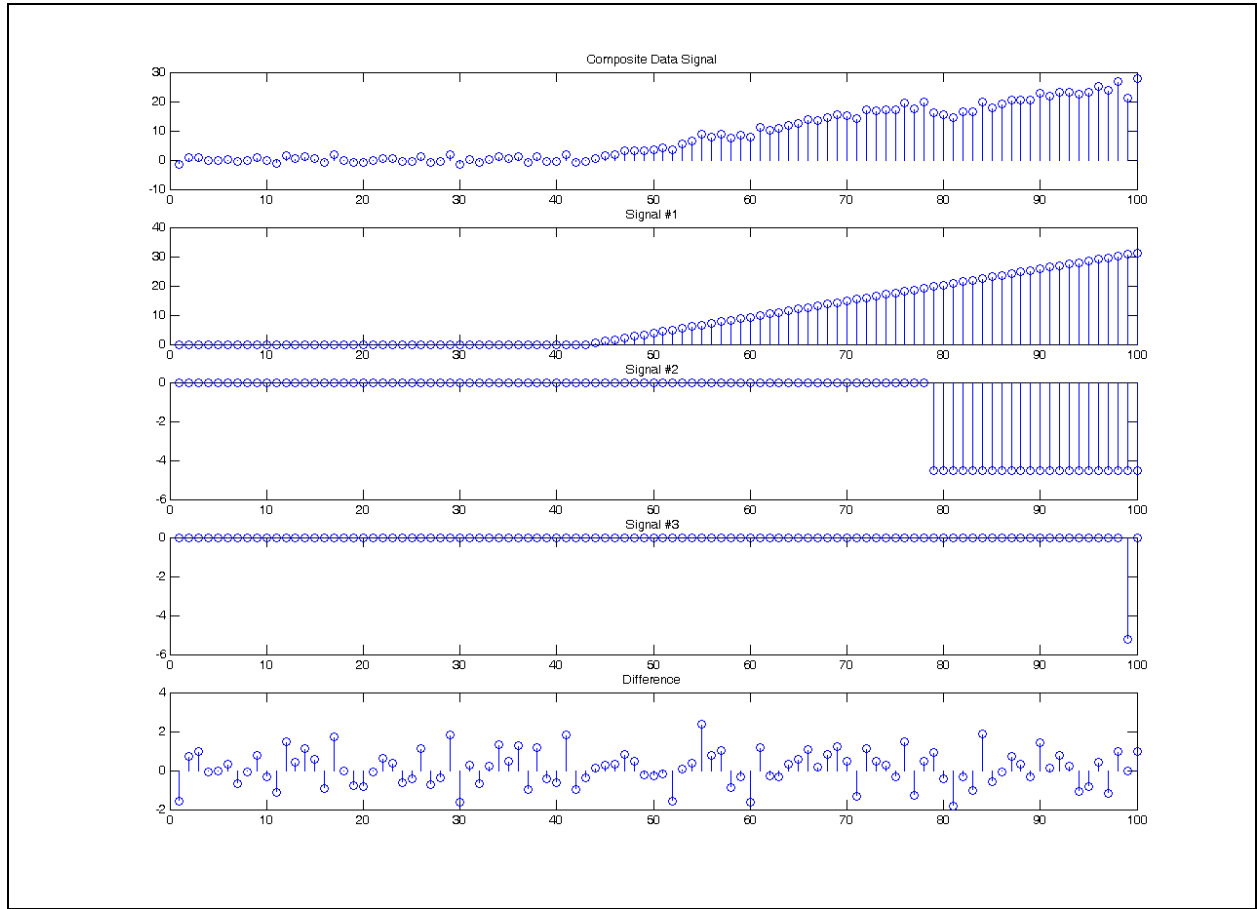
- 
- \* F.M.C.A. – Fast Minimum Complexity Algorithm
  - \*\* O.M.C.A. – Original Minimum Complexity Algorithm



**Figure 4.1 – Results of the F.M.C.A. on a Data Signal with No Additive Noise**

Figure 4.1 shows that each of the test signal components was captured, and looking at the error signal it appears that they were captured with a very low amount of error. Applying this algorithm with a window of 20 and a step size of 5 on a noisy signal with noise power of 1 yields similar results, shown in Figure 4.2. While not quite as fast as the algorithm when considering a window of 10, the extra data points in the window are important to combat the additive noise.





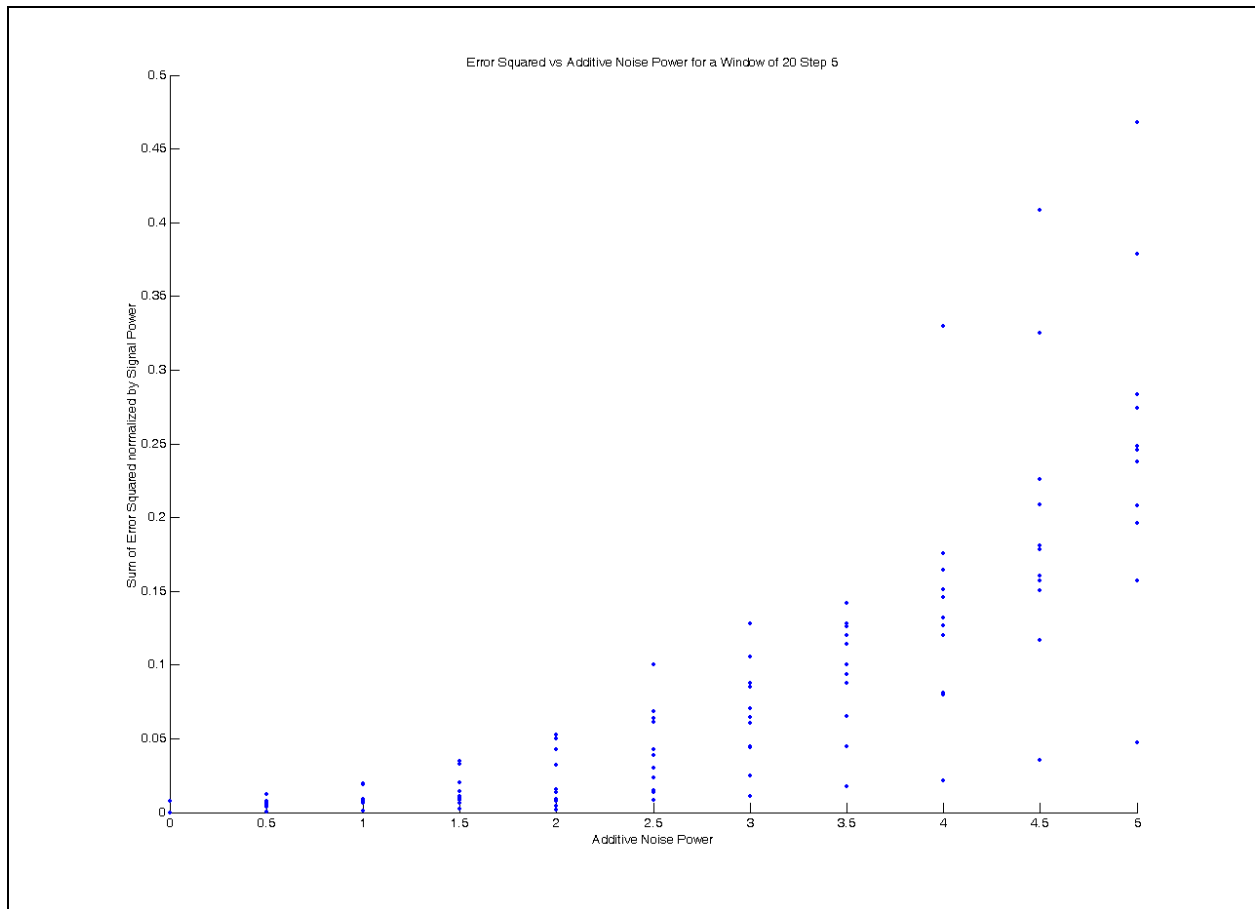
**Figure 4.2 – Results of the F.M.C.A. on a System With Additive Noise**

## 4.6 Quantitative Noise analysis

As of now, the original minimum complexity algorithm and the fast minimum complexity algorithm have been tested to see how close they can come to recovering the original components in a noisy signal. Even though the purpose of this paper was to find algorithms to generate the intuitive solution, for the sake of comparison, some quantitative measure must be employed. Because the intuitive solution is a very difficult quantity to define, as any solution given by a rational human could qualify as an intuitive solution, some means must be found to

characterize this solution. This author used a method in which easily recognizable signals were exposed to additive gaussian white noise. In this way the original components can be considered to be representative of the intuitive solution. In this way we can say that a good performance measure is comparing the original components to those detected by the algorithm. Finding such a comparison is surprisingly difficult. Because the desire is to measure how close an algorithm can come to an intuitive solution one must first consider what should the algorithms be penalized for. Many things must be considered, the algorithms should not only return the correct number of signals but also the correct types, magnitudes, and delays. Because these quantities express themselves in vastly different ways it is difficult to come up with an error measure that not only makes physical sense but also generates accurate comparisons. The following measurement is proposed as a possible error comparison, not necessarily the best comparison as different applications have different constraints.

In the plot below, the power of the error between the estimated signal and the actual signal is plotted against various additive noise powers. This essentially tests the noise reduction capability of the algorithm. Because of the numerous combinations of window size, threshold, and step size only a typical case is presented. A number of such tests were run on a fast minimum complexity algorithm with a window size of 20, a step size of 5, and a threshold of twice the noise power. The threshold was set purposefully high to achieve better performance at the higher noise levels. For better overall performance, as explained in the discussion of threshold, a moving threshold value is required.

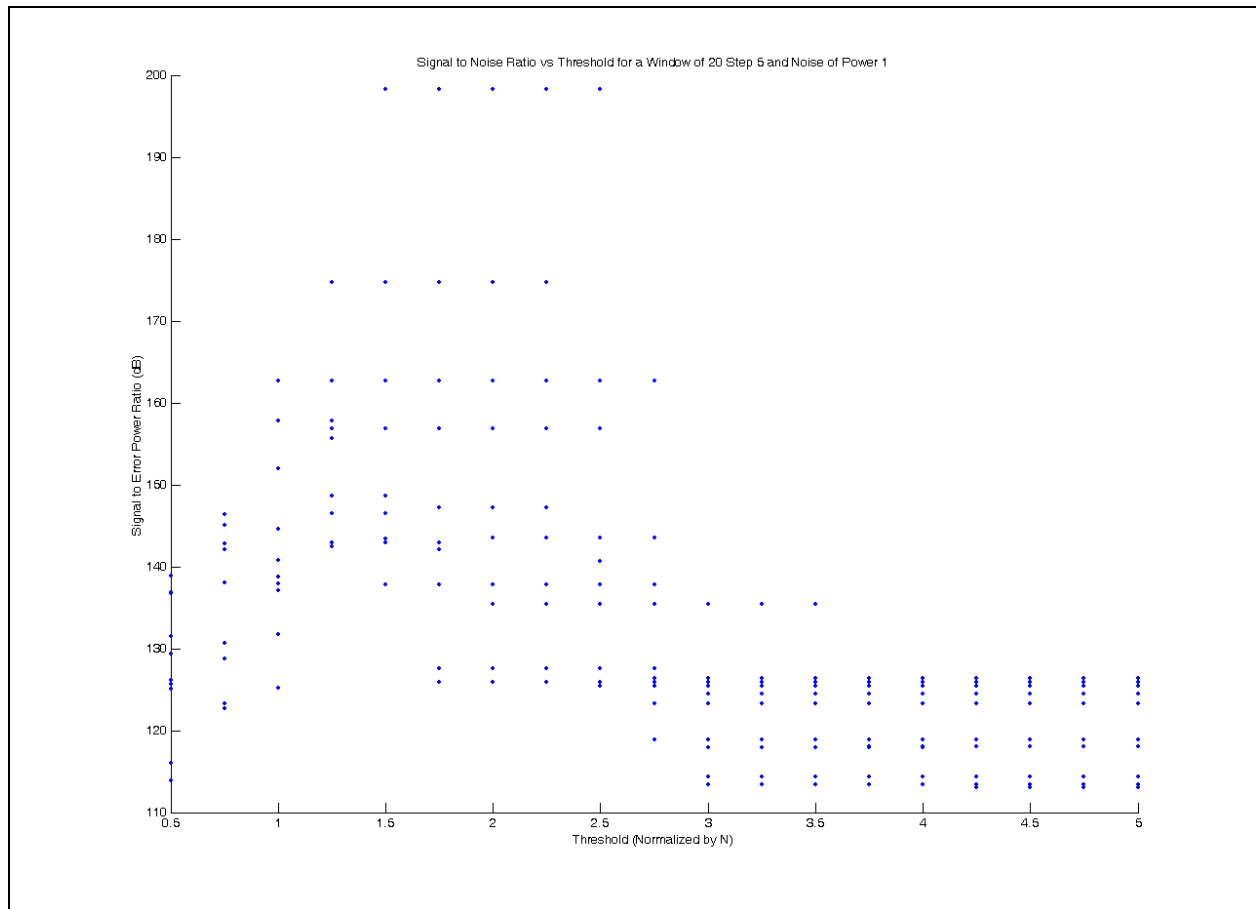


**Figure 4.3 – Relationship Between Noise Power and F.M.C.A. Performance**

The test shown in Figure 4.3 shows that the relationship between the error ratio and the additive noise power is either exponential or quadratic. The error is held to within five percent of the signal power up until the noise power approaches 2. While not always the case, on occasion the error was within five percent when the noise power was 5. In any case the error generally no more than twenty percent of the original signal.

#### 4.6.1 The Impact of the Threshold on Performance

The largest single impact on the performance of the system is the threshold. Because of the relationship between additive noise power, data length, and threshold, an improper threshold value can create problems. If the threshold is set too low, the algorithm will interpret noise as signals, if the threshold is set too high, the algorithm will fail to detect trends in the data. To illustrate this numerous tests were run on a composite signal composed of three randomly delayed and scaled component signals. A noise signal of power 1 was then added to this composite signal. For these tests the fast minimum complexity algorithm was run with different thresholds. Again because of the vast array of possible combinations of windows and step sizes, a window of length 20 and a step size of 5 were used to generate the plot given in Figure 4.4.

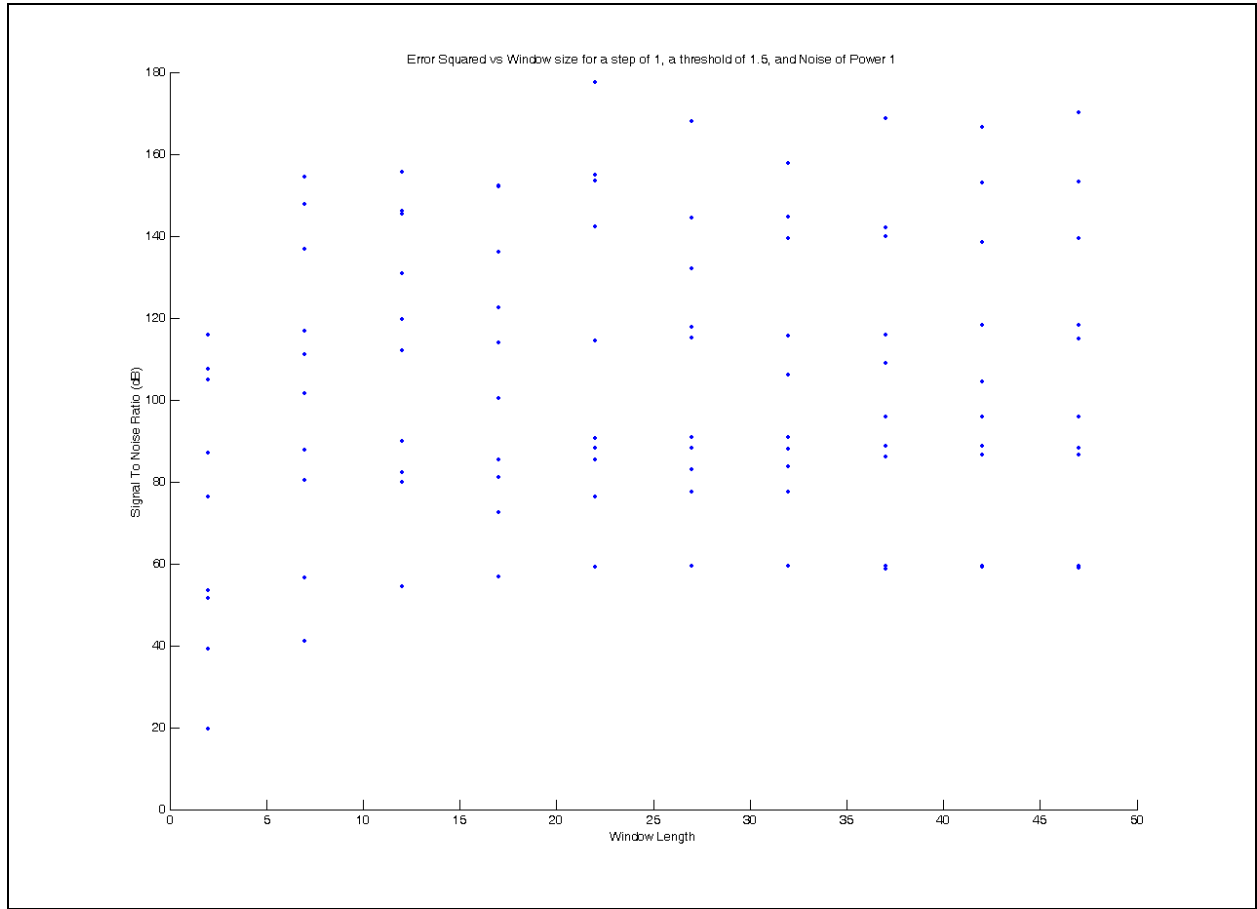


**Figure 4.4 – Signal to Noise Ratio for F.M.C.A vs Threshold**

After many trials, a clear trend is visible in Figure 4.4; the signal to noise ratio increases until the threshold reaches approximately 1.5. From there it slowly decreases until the threshold reaches about 3 where ratio levels off at around 120 dB. This shows the importance of selecting the correct threshold. The range of thresholds from 1.1 to about 2 generate a signal to noise ratio around 50 to 100 percent higher than other thresholds. Of course, due to the non-linear relationship between the threshold value and the additive noise power, higher threshold values and window lengths are required.

#### **4.6.2 The Effect of Changing the Window Length**

Window length is another important item. Selection of Window length is much more of a balancing act than threshold selection. For threshold selection, a certain region works best for a given additive noise power and window length. However, the longer a window length gets, the more accurate the algorithm should get. Unfortunately, that increase comes at a cost. Large window lengths can generate immense computational loads. Not only do longer window lengths have more data points, these data sets can also contain more component signals. These two factors working in conjunction can easily generate well in excess of one hundred times the computational load by merely doubling the window length. To illustrate the effect window length can have on the performance of the fast minimum complexity algorithm a test signal of length 100 made up of three composite signals and an additive noise vector of power 1 decomposed numerous times with various window lengths. To attempt to isolate solely the effect of the window length, the step size was relegated to one sample. Also to maintain uniformity with previous experiments the threshold was set at 2. The plot shown in Figure 4.5 catalogs the results of this testing.



**Figure 4.5 – Signal to Noise Ratio for F.M.C.A. vs Window Size**

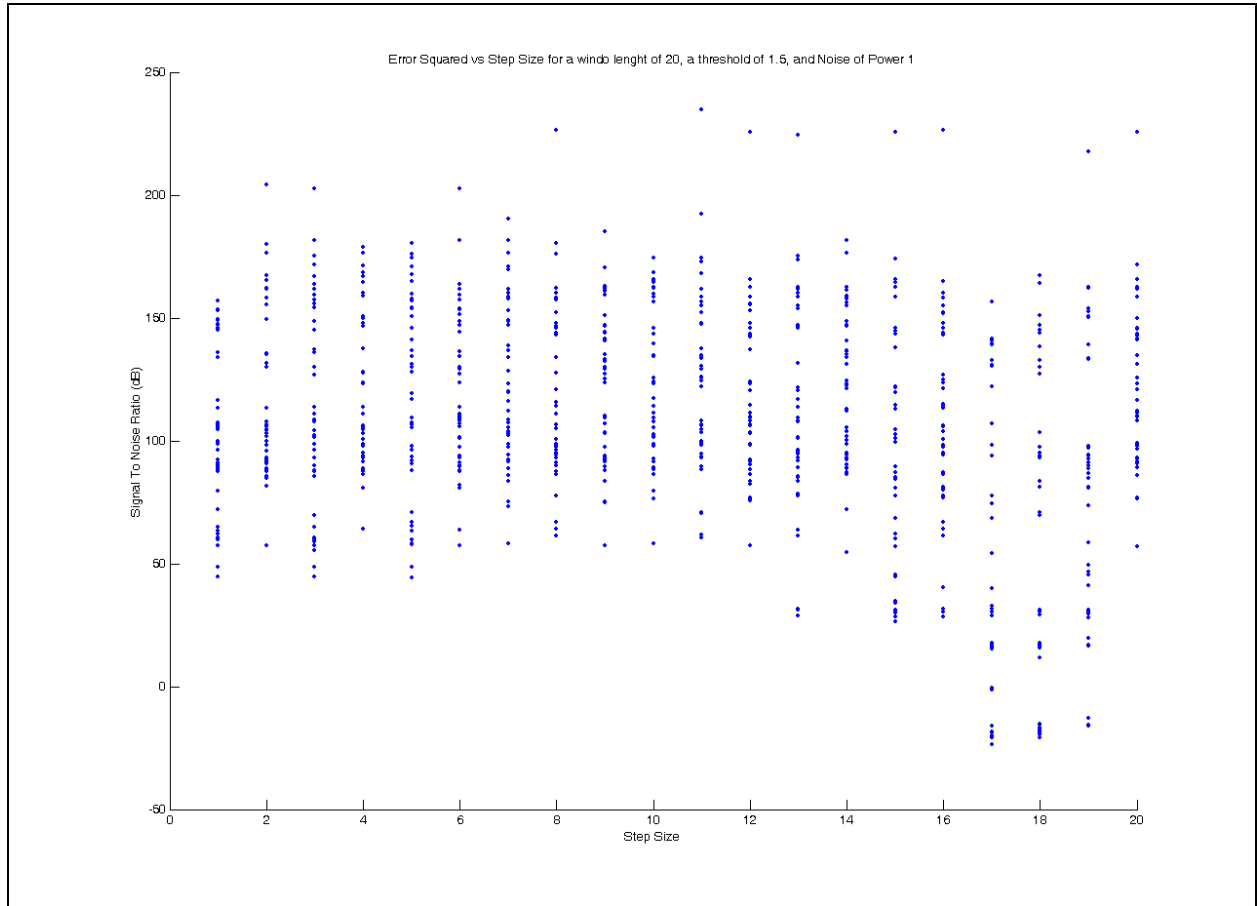
While not shown on the output of Figure 4.5, the most obvious result of the testing was the tremendous difference in speed between the small window lengths and the large window lengths. Using small window lengths, in the range of 15 to 20, the fast minimum complexity algorithm finished in less than 2 seconds, much faster in the case of smaller windows. However, when the window length approached 50, the algorithm took nearly 10 minutes to complete.

As expected, running the fast minimum complexity algorithm with larger window lengths produced much more accurate results than the tests run with small windows. However, despite

the large difference in computational time between the midsize windows and the large windows, there was little if any difference in performance. This is a result of two factors. The first is that due to time constraints the algorithm was limited to a maximum of two component signals in a given window. In addition, less and less new information is contained in a data signal the more data is added. The expected difference between averaging two and three values is much greater than when 49 or 50 samples are considered. The same phenomenon holds true with window sizes. For an additive noise power of 1, the best combination of accuracy and speed happens between 10 and 20 samples. In this range the accuracy is similar to that achieved with a window length in the 40's while taking a fraction of the time.

#### **4.6.3 The Relation Between Error and Step Size**

The final parameter to be considered is the step size. The changes not only the increment of the window, but also the minimum amount of data a possible component signal needs in front of it to be considered and actual component signal. This produces two effects, the first of which is speed. While not having near as large of an impact as window length, the size of the window increment linearly effects the duration of the algorithm. Double the increment and the computations will be reduced by a factor of 2. The other impact step size has is on accuracy. If the step size is too high then signals that should not be considered signals will be accepted. To illustrate this, consider the case of an impulse occurring at the last point in a window. Without any additional points it is not possible to say whether or not it is a step, a ramp, or an impulse. However, the algorithm will decide that it is a ramp, step or impulse. If that signal is within the increment, it will be stored as a component signal causing a great deal of error. This can be seen in the Figure 4.6.



**Figure 4.6 – Signal to Noise Ratio for F.M.C.A vs Window Increment**

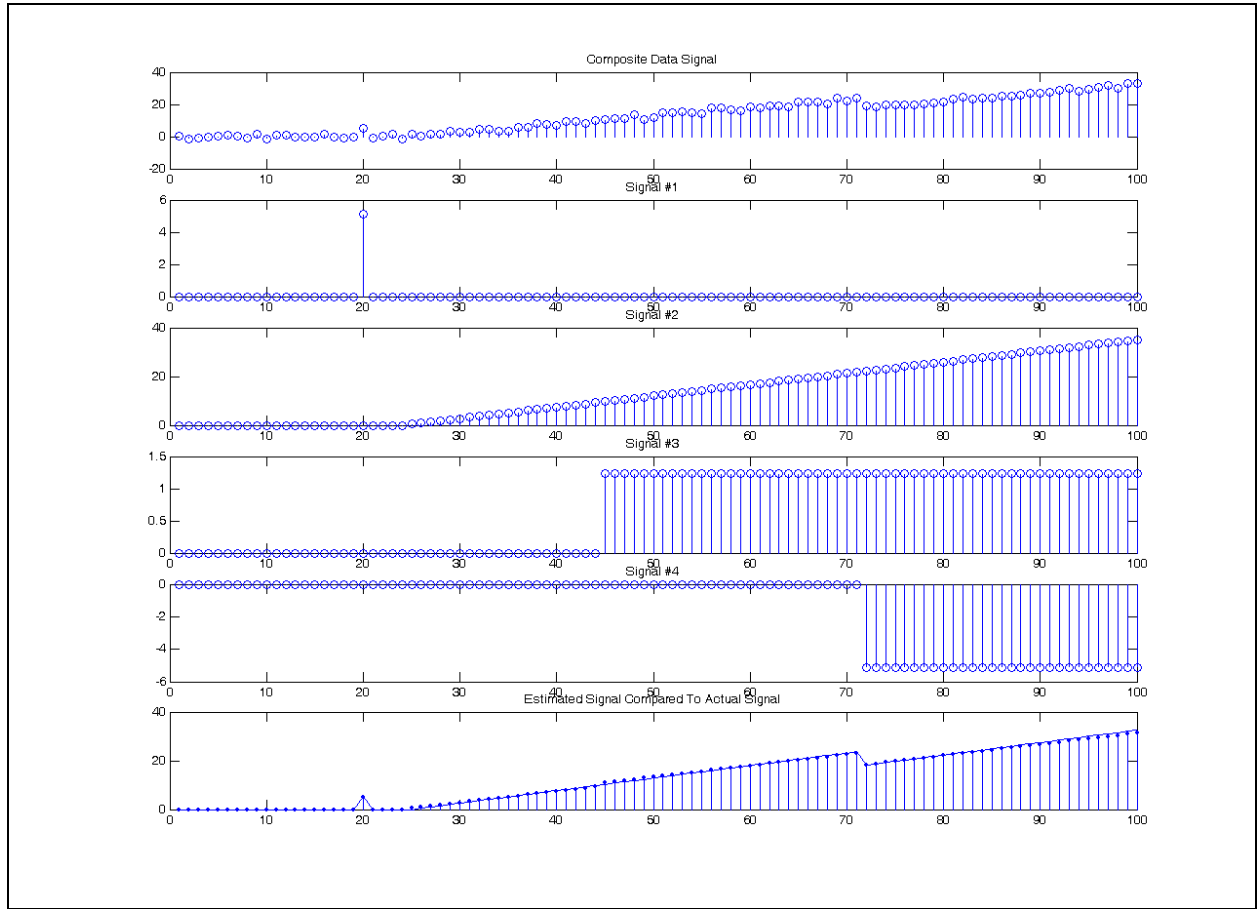
As expected if the step size is too large, that is larger than about half the window size, the algorithm behaves quite poorly, often boasting a signal to noise ratio less than 0 db. Step sizes in the region of one half the window size and less seem to give the best results. Although the plot in Figure 4.6 again suffers from the problems caused by limiting the possible number of component signals to 2 over a given window, the algorithm seems to work quite well with signal to noise ratio usually greater than 100 dB.



## 4.7 Conclusions

With properly balanced selections of the threshold, the window length, and the step size, the fast minimum complexity algorithm is capable of accuracy approximating that of the original minimum complexity algorithm while at a fraction of the computing cost. While the optimal quantities vary from system to system, fortunately in typical systems composed of ramps, steps, and impulses general starting point exists. Setting the threshold to 1.5 times the approximate additive noise power, the window length to roughly 15 to 20 data points, setting the window increment to 5 usually generates good results. A higher threshold can be used to combat high noise levels as well as longer window lengths and shorter window increments. If the component signals are very close together, decreasing the window size can help speed up the algorithm.

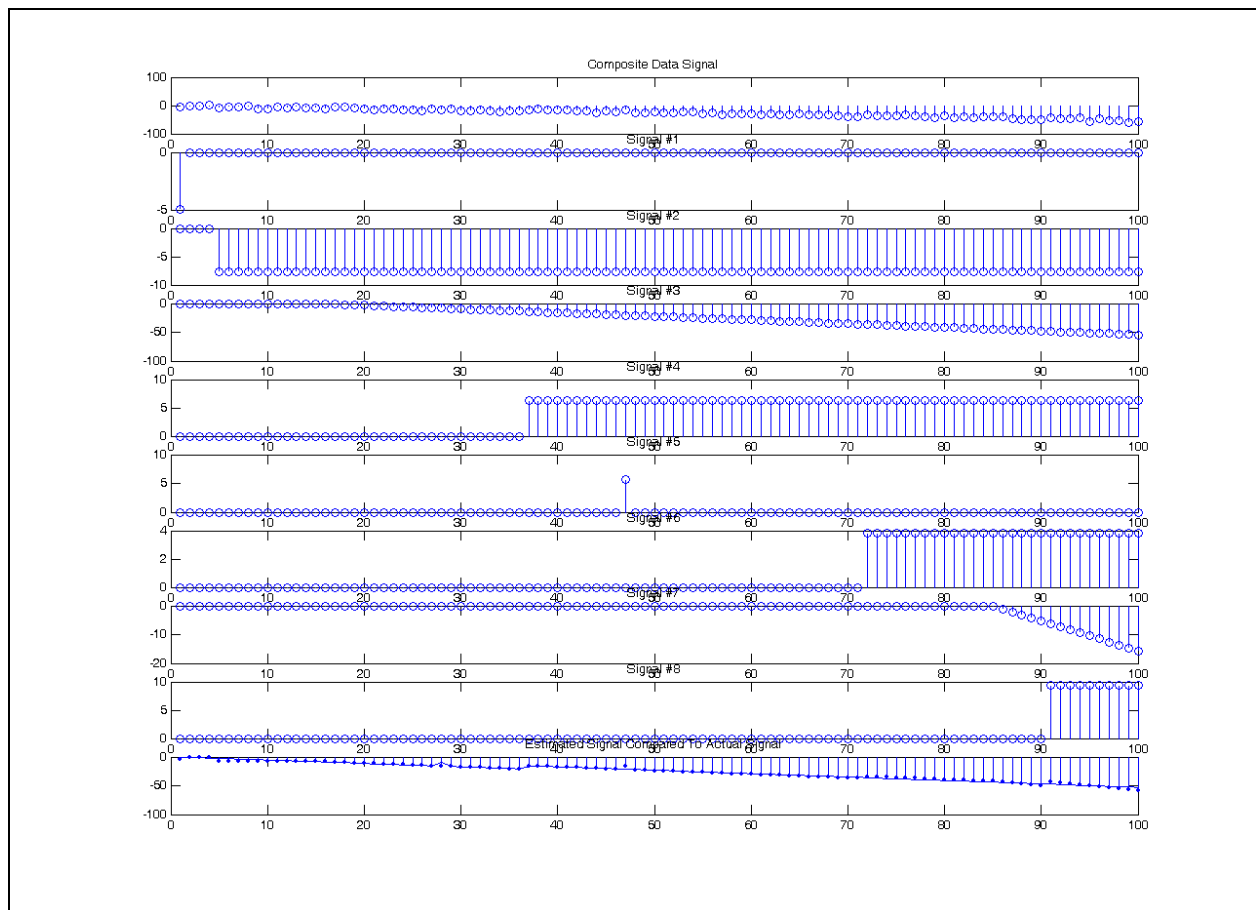
With a minimal amount of fine-tuning, the fast minimum complexity algorithm proves to be quite accurate. Even when considering moderately noisy signals, few false signals are reported as shown by the graph in Figure 4.7.



**Figure 4.7 – F.M.C.A. Applied to Noisy Signal**

While the composite signal shown in Figure 4.7 was composed of three component signals, an impulse, a ramp and a step, the algorithm returned an impulse, a ramp, and two steps. Considering that the false signal has a very small magnitude and therefore would easily be rejected as a false signal, the fast minimum complexity algorithm can claim near perfect signal recovery, as shown by the last plot in Figure 4.7. In that plot the estimated signal is laid over the original signal, showing that they are nearly identical. In addition to the performance of the algorithm, with proper post analysis of the results of this algorithm, false signals can be almost

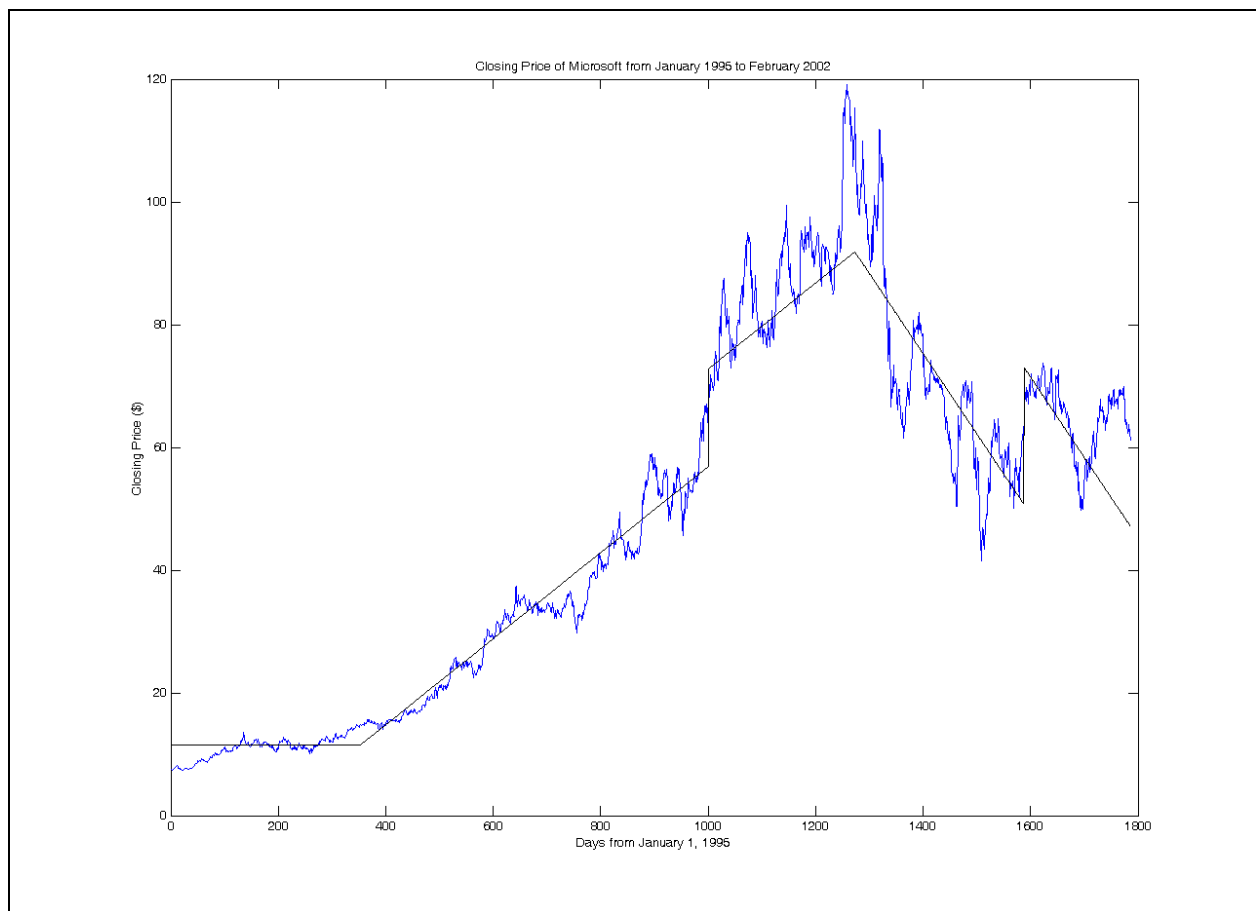
completely eliminated. This is not to say that this algorithm never breaks down. At very high noise levels the algorithm breaks down, returning too many component signals.



**Figure 4.8 – F.M.C.A Applied to Very Noisy Signal**

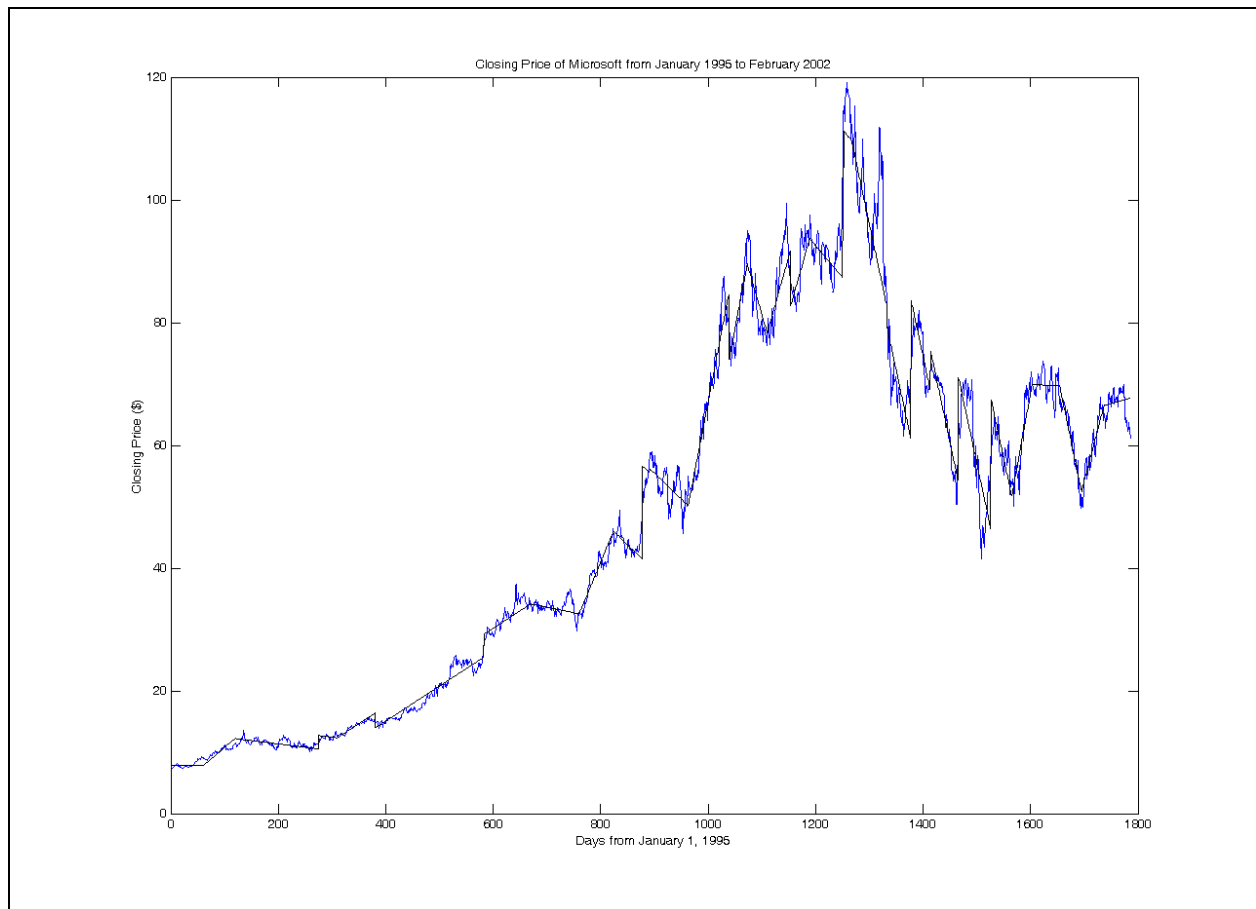
The algorithm returns too many signals because with high levels of noise, large amounts of data are required between individual signals. If the noise gets too high, the amount of data required to detect a signal correctly becomes greater than the space between the individual component signals, causing false signal detection.

The most important criteria in determining if the solution generated by an algorithm matches the intuitive solution is how closely the human eye agrees with the results. As a final test for the minimum energy approach, the algorithm was applied to the Microsoft data discussed earlier. For the first test, shown in Figure 4.9, the window was set to a very large value, 400 days. Also the noise tolerance was set very high. This is intended to simulate a very rough scan through the data.



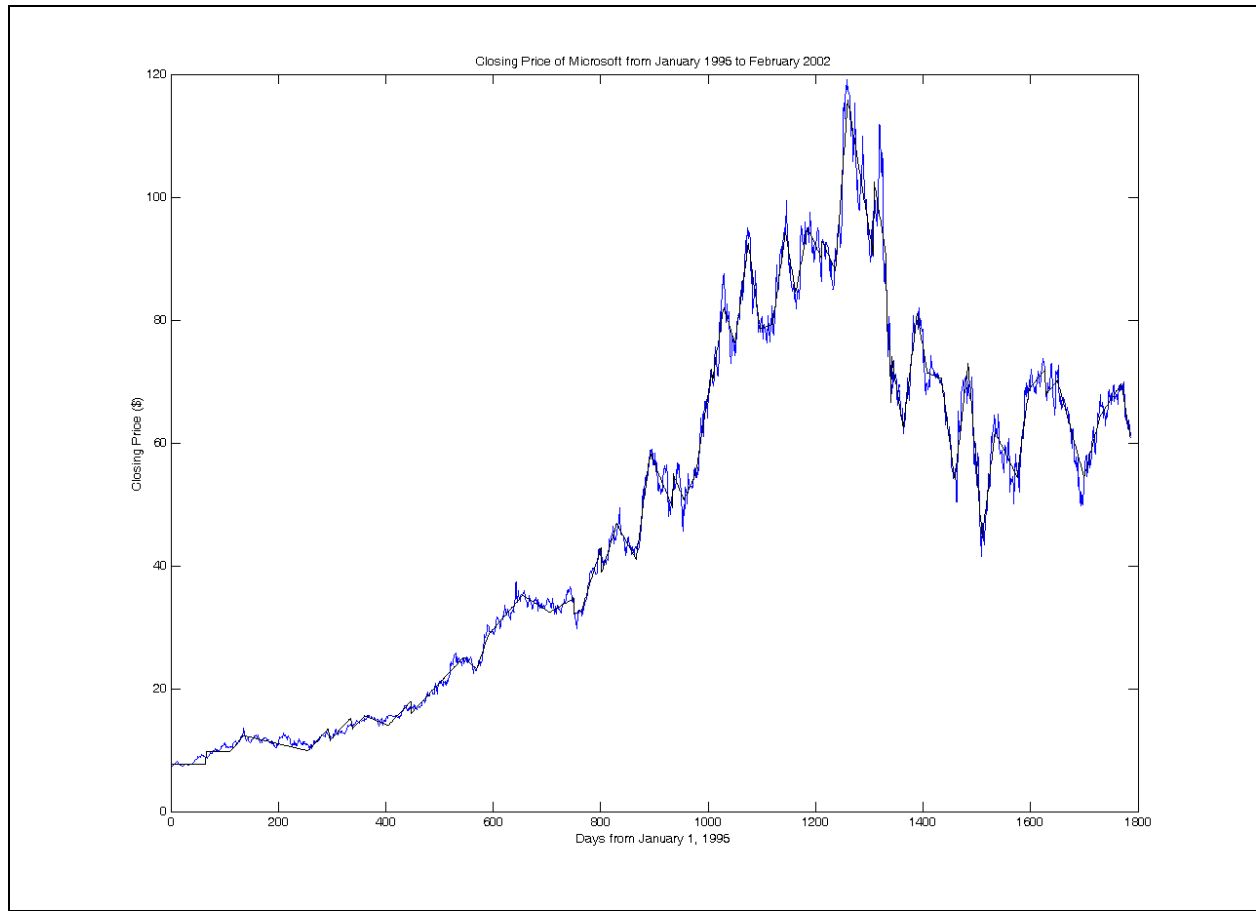
**Figure 4.9 – FMCA Approximation of Microsoft Share Price (Long Term Trends)**

The algorithm appears to agree with what the eye expects. Of course this is not the only representation of the signal. A user might desire a more detailed approximation of the price. To simulate this the window of the fast minimum complexity algorithm as well as the error tolerance were reduced, resulting in the Figure 4.10.



**Figure 4.10 - FMCA Approximation of Microsoft Share Price (Medium Length Trends)**

Once again it appears that the fast minimum complexity algorithm has captured an acceptable medium length approximation of the data. Finally, when considering very short length trends, the algorithm again proves to be very accurate at selecting the desired intuitive solution.



**Figure 4.11 - FMCA Approximation of Microsoft Share Price (Short Length Trends)**

Therefore, we can say with relative certainty that the minimum complexity algorithm consistently produces an accurate representation of the intuitive solution. Although the intuitive

solution changes based on the application, by adjusting the parameters of the algorithm, the required intuitive solution can be found.

While very effective, this algorithm is not perfect. If a method was found to decrease the number of possible combinations of signals and delays, it could greatly decrease the computational time need to run this algorithm. With such an increase in speed it would then be possible to use larger window sizes and consider larger numbers of signals per window, improving the accuracy of the algorithm. However, the fast minimum complexity algorithm does provide the best balance of speed and accuracy of all of the methods studied in this paper.

## **5.0 Final Comparison and Conclusion**

### **5.1 End Goal of Paper**

The human eye is innately able to separate a signal into a combination of known signal classes of primitive signals. In many applications, however, there is no human in the decision loop and hence this process must be automated. Also, situations arise where it is desirable to obtain an accurate decomposition of a signal but noise, time constraints, and/or complexity prohibits the human eye from effectively recognizing the intuitive solution. Again, some sort of automated method of generating this solution is desired. However, before an automated approach to this problem can be devised, the intuitive solution must be mathematically explained.

This paper examines algorithms based on two possible mathematical requirements that an algorithm must satisfy to yield the intuitive signal decomposition. These two requirements are that of minimum energy and minimum complexity. Algorithms based on these requirements were introduced and characterized. The results of the algorithms based on these two requirements strongly suggest that the intuitive solutions are best obtained using the minimum complexity approach. That is to say that is the signal decomposition containing the fewest number of component signals is the most desirable result, regardless of the energy involved.

### **5.2 Thoughts on Minimum Energy Approach**

While the minimum energy approach failed to generate the intuitive results, it was not without some merit. This method proved to be considerably faster than the algorithms based on minimum complexity. It also yielded results similar to that of the intuitive solution when applied to data sets containing very low levels of correlation between component signal types. However, when applied to signals with even moderate amounts of cross correlation, the results bear little



resemblance to the intuitive solution. It is assumed that for such an approach to yield the intuitive solution on all signals, the cross correlation between the individual component signal types must be accounted for.

### **5.3 Thoughts on Minimum Complexity Approach**

Algorithms based on the minimum complexity approach constantly generated solutions consistent with the intuitive solution. This seems to imply the validity of considering the minimum complexity result equivalent to the intuitive solution. Unfortunately, it appears that a great deal of computational power is required to compute the minimum complexity solution. To help alleviate some of the computational load, an approximation to this algorithm was presented. When properly implemented, this approximation yields results equivalent to the original algorithm at a greatly reduced computational cost. The increase in speed as well as the relationship between the parameters of this fast approximation algorithm and the data were also considered. Finally, general rules for the selection of these parameters were presented.

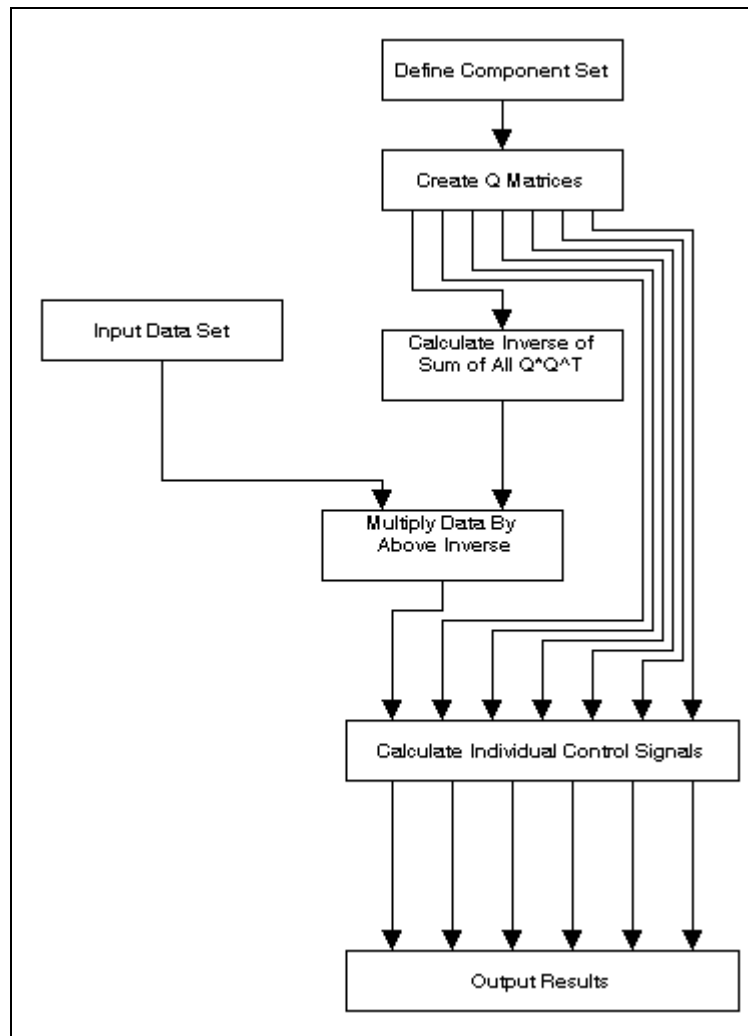
Many uses can be found for such an algorithm. For example event regions can be defined when using data sets such as the impulse, ramp, step. In the case of many impulse, ramp, step systems a ramp will occur and then level off. Also instead of a simply a step, the data will contain pulses of data. The regions over which these occur can be thought of as events, such as in the case of troop deployment. The period over which the number of troops in a certain area is increasing, or pulsed up or down can be a very noteworthy time period. If the minimum complexity algorithm is applied to such a system, the resulting component signals will have the characteristic that detected signals will usually be followed by signals of equal type but of nearly opposite magnitude. Thus by looking the resulting component signals, it is fairly easy to recover and classify these regions into events.

## 5.4 Future Possibilities

Though a working method of signal decomposition into primitive classes was presented in this thesis, better methods may still exist. For example instead of looking at a minimum energy approach, perhaps better results can be had with a minimum entropy approach. This criteria might generate a much more peaky set of control signals. In the case of the minimum complexity approach, it might not be necessary to look at every possible signal delay combination. Points where the signal does not drastically change might be eliminated, reducing computational time and permitting larger window sizes larger numbers of component signals.

## Appendix A – Algorithm Block Diagrams

### Minimum Energy Approach



**Figure A.1 – Minimum Energy Algorithm Flow Chart**

## Original Minimum Complexity Algorithm

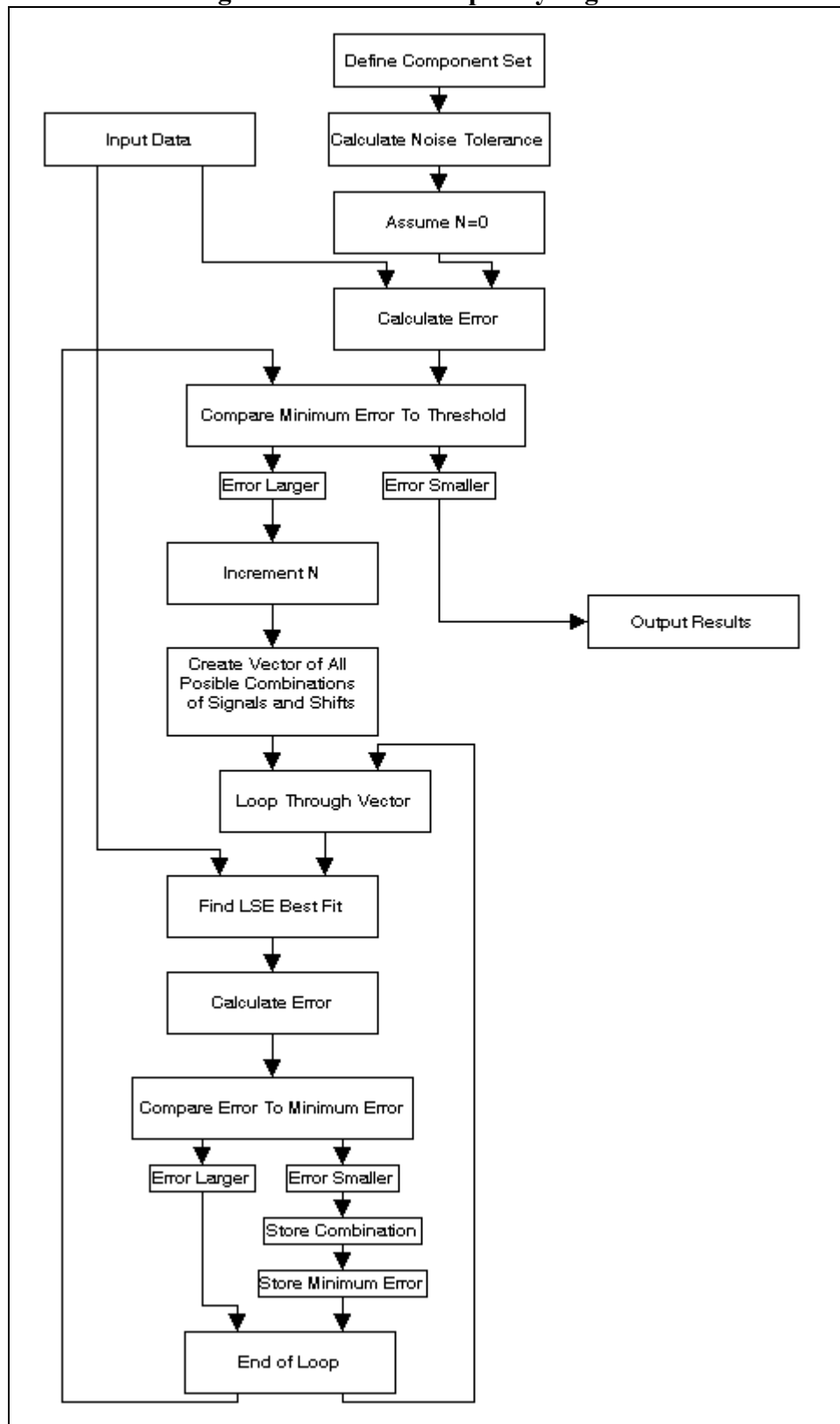


Figure A.2 – Minimum Complexity Algorithm Flow Chart

## Fast Minimum Complexity Algorithm

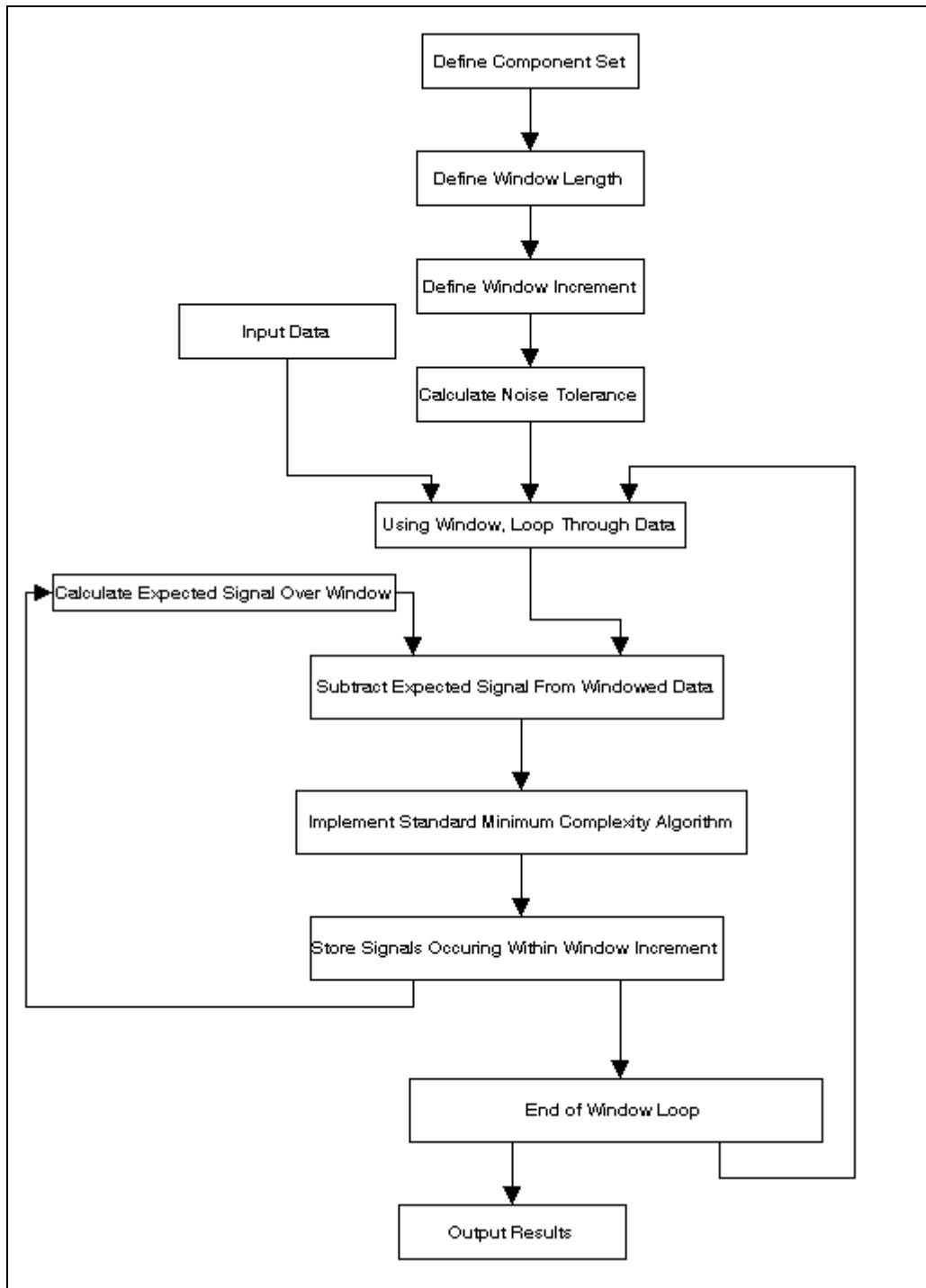


Figure A.3 – Fast Minimum Complexity Algorithm Flow Chart

## **Appendix B – Sample Matlab Code**

To Further Aid the reader in understanding possible methods of implementing the algorithms discussed in this paper, an appendix comprised of examples of Matlab code has been included. The following sections contain the actual code used to achieve the results presented in the aforementioned examples. These sections are organized by algorithm and contain the scripts and functions used. In a few examples, particularly in the case of the fast minimum complexity algorithm, code used but also given in a previous section has been omitted.

## Minimum Energy Approach

### Example Minimum Energy Algorithm

```
% Minimum Energy Multisignal Deconvolution
% Variable initialization

N = 50;
time = 1:N;
siglev = 5;
sigdev = 2;
sigs = [1 zeros(1,N-1); ones(1,N); cumsum(ones(1,N))/5];
nsigs = 3;

% create composite signal
composite = zeros(N,1);

xeng = 0;
for c1 = 1:nsigs
    type = c1;%floor(size(sigs,1)*rand(1) + 1)
    sigmag = (2*floor(rand(1)+.5)-1)*(abs(sigdev*randn(1)+.5)+siglev)
    delay = floor(N*rand(1))
    xeng = sigmag^2+xeng;
    tempsig = sigmag*[zeros(1,delay) sigs(type,1:N-delay)];

    composite = composite + tempsig;
end

% create Q (matrix convolution) Matrices
Q1 = convq(sigs(1,:))';
Q2 = convq(sigs(2,:))';
Q3 = convq(sigs(3,:))';

% find sum of matrix

matsum = Q1*Q1' + Q2*Q2' + Q3*Q3';
matsumi = inv(matsum);

% find control Signals
mulfact = matsumi*composite;

s1 = Q1' * mulfact;
```

```
s2 = Q2' * mulfact;  
s3 = Q3' * mulfact;
```



## Random Composite Signal Generator

```
function [dsr] = randomdsrgen(N,nd,ns,nr,min,var,noise)

% random signal generator of signals made up of ramps steps and deltas
% output = randomdsrgen(# Samples,# deltas,#steps,
#ramps,minvalue,var,noiselevel);
%

% generate random points
numd = floor(rand(1,nd)*N+.5);
nums = floor(rand(1,ns)*N+.5);
numr = floor(rand(1,nr)*N+.5);

% generate marker signals
md = zeros(1,N);
ms = zeros(1,N);
mr = zeros(1,N);

for counter = 1:numd
    md(numd(counter)) = md(numd(counter)) + rmarkgen(min,var);
end

for counter = 1:nums
    ms(nums(counter)) = md(nums(counter)) + rmarkgen(min,var);
end

for counter = 1:numr
    mr(numr(counter)) = md(numr(counter)) + rmarkgen(min,var);
end
mr = mr/N;

% generate output signal
dsr = md + cumsum(ms) + cumsum(cumsum(mr)) + randn(1,N)*(noise^.5);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [rmarker] = rmarkgen(min,var);

rmarker = ((var)^.5*abs(randn)+min)*(2*(floor(rand+.5)-.5));
```

### Q Matrix Utility

```
function [q] = convq(s);

% Convolution Matrix Generator
%
% Will generate a matrix Q from s such that conv(s,x) = s*Q where s is a 1*N matrix
% q=convq(s);

N = length(s);

q = zeros(N,N);

for c1 = 1:N

    for c2 = c1:N

        q(c1,c2) = s(c2-c1+1);

    end

end
```

## Original Minimum Complexity Algorithm

### Example Minimum Complexity Algorithm

```

function [minshifts, mina,n] = decon(y,noiselev,maxn);

% [minshifts, mina,n] = decon(y,noiselev,maxn);
% minshifts = vector of signals
% mina = vector of cooresponding magnitudes
% n is the number of signals
% noiselev is estimated noise power
% maxn is highest level n allowed
%
%Demonstration of minimum complexity multisignal deconvolution algorithm
% created 01/07/02 by Dave Galati

%Variable Initialization
N = length(y); % Size of Signal
signo = 3;
sigs = [1 zeros(1,N-1) ;ones(1,N); cumsum((ones(1,N)))];

mina = 0;
minshifts = 0;
minerror = 0;
mincomp = zeros(1,N);

% starting algorithm
minerror = (norm(y))^2;
n = 0; % Total Number of Signals

while (minerror) > .001 + noiselev*N & (maxn > n);

    % increment number of signals
    n = n+1;

    % Make vector of all posible signals;
    delays = 0:N-1;
    siglist = null(1);
    for c1 = 1:signo
        siglist = [siglist (delays+1000*c1)];
    end

    % create matrix of all posible combos
    combos = nchoosek(siglist,n);
    % innitalize shifts

```

```

shifts=      zeros(2,n);

% shift through shifts variable
for c1      =      1:size(combos,1);

    % create shifts variable;
    for c2      =      1:n
        shifts(1,c2)      =      floor(combos(c1,c2)/1000);
        shifts(2,c2)      =      combos(c1,c2) - 1000*shifts(1,c2);
    end;

    comp      =      compcreate(shifts,sigs);
    [a,error]= lse2(comp,y);

    if (error < minerror)
        mina      =      a;
        minshifts=      shifts;
        minerror      =      error;
        mincomp      =      comp;
    end;

end;

end

```

### Example Least Squared Error Approximation

```
function [a,error] = lse2(components,y);

N=size(components,2);

% calculate cov matrix
h = zeros(N,N);
for c1 = 1:N
    for c2= 1:N

        h(c1,c2) = components(:,c1)*components(:,c2);

    end
end

% calculate cov between output and each signal
z = zeros(1,N);
for c1 = 1:N

    z(c1)= y*components(:,c1);

end

% calculate a where a*h=cov(y,components)
a = pinv(h)*z';
error=(norm((y'-components*a)))^2;
```

### Component Matrix Utility

```
function [comp] = compcreate(shifts,sigs);

% [comp] = compcreate(shifts,sigs)
% creates a component matrix for use in lse
% shifts is a 2*N vector where 1st row is signal and second row is shifts
% sigs is a vector of signals where each row is a different signal

% find signal length
sigl = size(sigs,2);

% find number of signals
N = size(shifts,2);

% create output;
comp = zeros(N,sigl);
for c1 = 1:N;

    temp = [zeros(1,shifts(2,c1)) sigs(shifts(1,c1),1:sigl-shifts(2,c1))];
    for c2 = 1:sigl
        comp(c1,c2) = temp(c2);
    end

end

comp = comp';
```

## Fast Minimum Complexity Algorithm

### Example Fast Minimum Complexity Algorithm

```
function [shifts,mina] = deconscan(y,winlen,stplen,noiselev,maxn);
%
%
% fast minimum complexity multisignal deconvolution algorithm
%
% [shifts mina] = deconscan(y,winlen,stplen,noiselev,maxn)
%
% shifts = shifts class signal (see compcreate)
% mina = magnitude of signals vector
% y = data signal
% winlen = window length
% stplen = step size
% noiselev = noise level
% maxn = max number of signal per window.

%Variable Initialization
N = length(y); % Size of Signal

% algorithm implimentation

% insure an integer number of steps
N = floor(N/stplen)*stplen;
y = y(1:N);

% initialize loop
sigavg = 0; %Signal Average;
sigslp = 0; %Signal slope;
slphlp = 0:winlen-1; %Slope helper
shifts = null(1);
mina = null(1);
counter = 0;

% algorithm loop
for c1 = 1:stplen:N-winlen+1;

    % increment the signal average
    sigavg = sigavg + sigslp * stplen;
```

```

% read in temporary y
ytemp = y(c1:c1+winlen-1) - (sigavg + sigslp*slphlp);

% mcm decon of signal;
[tshifts, tmina, n] = decon(ytemp,noiselev*1.3,maxn);

% Store signals within the step size
for c2 = 1:n;
    if tshifts(2,c2) <= stplen;
        counter = counter+1;
        shifts(1,counter) = tshifts(1,c2);
        shifts(2,counter) = c1 - 1 + tshifts(2,c2);
        mina(counter) = tmina(c2);

        if tshifts(1,c2) == 2
            sigavg = sigavg + tmina(c2);
        elseif tshifts(1,c2) == 3
            sigslp = sigslp + tmina(c2);
            sigavg = sigavg + tmina(c2)*(1-tshifts(2,c2));
        end
    end
end
end

% Store final group of signals not included in the step size.
for c3 = 1:n;
    if tshifts(2,c3) > stplen;
        counter = counter+1;
        shifts(1,counter) = tshifts(1,c3);
        shifts(2,counter) = c1 - 1 + tshifts(2,c3);
        mina(counter) = tmina(c3);
    end
end
end

```



## Bibliography

- [1] de Prony (Gaspard Riche), Baron R. "Essai Experimental et Analytique: sur les Loix de la Diatabilite de Fluides Elastiques et Sur Cellas de la Force Expansive De la Vapeur de l'Eau et de la Vapeur de l'Alcool, a Differentes Temperatures," *Journal de l'Ecole Polytechnique, Paris* **1**:2, pp. 24-76, 1775.
- [2] Trench, W. "An Algorithm for the Inversion of Finite Toeplitz Matrices," *SIAM J Appl Math* **12**:3, pp. 515-522, September 1964.
- [3] Zohar, S. "The Solution of a Toeplitz Set of Linear Equations" *J Ass Comput Mach* **21**, pp. 272-276, April 1974
- [4] Zohar, S. "Toeplitz Matrix Inversion: The Algorithm of W.F.Trench" *J Ass Comput Mach* **16**, pp. 592-601, October 1969.
- [5] Hildebrand, F.B., *Introduction to Numerical Analysis*. p. 379, McGraw Hill, 1956.
- [6] Scharf, Louis L., *Statistical Signal Processing*. Addison-Wesley Publ. Comp., 1990
- [7] Tufts, D., and R.Kumaresan, "Frequency Estimation of Multiple Sinusoids: Making Linear Prediction like Maximum Likelihood," *Proc IEEE* **70**, pp. 975-990, March 1983
- [8] Grenader, O., and s. Szego, *Toeplitz Forms and their Applications* University of California Press, 1958
- [9] Pisarenko, V.F. "The Retrieval of Harmonics from a Covariance Function" *Geophysics Journal Royal Astronomical Soc* **33**,pp. 347-366, 1973.
- [10] Chan, Y. T., J. M. Lavoie, and J. B. Plant, "A Parametric Estimation Approach to Estimation of Frequencies of Sinusoids," *IEEE Trans ASSP* **29**:2, pp 214-219, April 1981.
- [11] Schmidt, R. O. "A Signal Subspace Approach to Multiple Emmtier Location and Spectral Estimation," Ph.D. dissertation, Stanford University, 1981.
- [12] Luenberger, David G., *Optimization by Vector Space Methods*. John Wiley and Sons Inc, 1969
- [13] Blahut, Richard E., *Fast Algorithms for Digital Signal Processing*. Addison-Wesley Publ. Comp., 1985.
- [14] Chappiro, L.E., "Statistical Signal Proccesing Notes".2001
- [15] Cadzow, James A., *Foundations of Digital Signal Processign and Data Analysis*. Macmillan Publishing Comp., 1987

- [16] Cruz, J.B., M.A. Simaan, A. Gacic, H. Jiang, B. Letellier, M. Li, Y. Liu, “Game-Theoretic Modeling and Control of a Military Air Operation,” *IEEE Trans AES* **37:4** pp 1393-1403